1541 DİSKET SÜRÜCÜ

KULLANIM KILAVUZU

Commodore

COMPUTER

Teleteknik

ELEKTRONİK SANAYİ VE TİCARET A.Ş.

Merkez: Rıhtım Cad. Nesli Han 207/2 Karaköy 80030 İstanbul
Tel: 1-152 50 33 (6 hat) Fax: 1-151 84 56 Tlx: 25731 tltk tr.
Sistem ve Hizmet Satış Bölümü: Silahhane Cad. Ralli Apt. 59/1
Teşvikiye 80200 İstanbul Tel: 1-131 15 49-50 Fax: 1-148 19 12
Ankara Bürosu: Çankaya Cad. No.10/5 Çankaya 06680 Ankara
Tel: 4-140 68 90 (3 hat) Fax: 4-140 68 80 Tlx: 46749 tktl tr.

İÇİNDEKİLER

1. GENEL TANITIM

2. KUTUNUN AÇILMASI VE BAĞLANTILAR 7
Kutunun İçindekiler 7
Kabloların Bağlanması 7
Çalıştırma 7
Disketin Takılması 7
VIC 20, COMMODORE 64,C - 16 veya + 4 ile kullanım 10
Disketlerinizi tanıyın 10

3. PROGRAMLARI KULLANMA 11
Hazır yazılımın yüklenmesi 11
LOAD 11
Disk Kataloğu 12
Örnek Eşleme ve Jokerler 13
SAVE 14
SAVE ve değiştirme 14
VERIFY 14
Dos Testek Programı 14

4. DİSK KOMUTLARI 15
OPEN ve PRINT 15
NEW 16
COPY 16
RENAME 17
SCRATCH 17
INITIALIZE 17
VALIDATE 17
Hata Kanalının Okunması 18
CLOSE 18

5. SIRASAL DOSYALAR 19
OPEN 19
PRINT ve INPUT 19
GET 21
Kataloğun Okunması 22

6. RASTGELE ERİŞİM 26
Rastgele erişim verileri için bir kanal açılması 27
BLOCK - READ 27
BLOCK - WRITE 27
BLOCK - ALLOCATE 28
BLOCK - FREE 29
BUFFER - POINTER 30
USER 1 ve USER 2 31

7. RELATİF DOSYALAR 32
Relatif dosyanın yaratılması 32
Relatif dosyaların kullanılması 33
Bir relatif dosya yaratın 38
Özet 38
Programın değiştirilmesi 39

8. DİSK KONTROLUNUN PROGRAMLANMASI 39
BLOCK - EXECUTE 39
MEMORY - READ 39
MEMORY - WRITE 40
MEMORY - EXECUTE 40
USER Komutları 41

9. DİSK CİHAZ NUMARASININ DEĞİŞTİRİLMESİ 41
Yazılım Metodu 41
Donanım Metodu 42

10. ASSEMBLY DİLİ VE 1541 42

EKLER:
A. Disk Komutları Özeti 44
B. Hata mesajları 44

1. GENEL TANITIM

GİRİŞ

Commodore 64 bilgisayarınız için mevcut en hızlı, en kolay ve en verimli dosyalama sistemine, 1541 Disket Sürücü'nüze hoş geldiniz. Bu el kitabı, acemi veya profesyonel olmanızdan bağımsız olarak disket sürücünüzden azami verimi nasıl alabileceğinizi göstermek üzere yazılmıştır.

Eğer acemi iseniz, ilk birkaç bölüm size disk sürücünün kurulması ve çalıştırılması hakkında temel bilgiler verecektir. Yeteneğiniz ve programlama bilginiz geliştikçe, disket sürücünüz için daha fazla kullanım alanı bulacağınızdan, bu el kitabının daha ileri bölümleri sizin için daha da değerlenecektir.

Eğer profesyonel iseniz, bu kılavuz size, düşünebildiğiniz tüm disket sürücü işleri için 1541'i çalıştırmanın yollarını gösterecektir.

Uzmanlık seviyeniz ne olursa olsun, 1541 disk sürücünüz, bilgisayar sisteminizin kapasitesini genelde fazlasıyla arttıracaktır.

1541'in çalışmasının detaylarına inmeden önce, birkaç önemli noktayı görmeniz gerekir. Bu el kitabı bir KILAVUZ'dur, yani aradığınız bilgi doğrudan disk ve disk sürücü ile ilgili olmadıkça, programlama bilgisi için Commodore 64 Kullanım ve Programcının El Kitabı'na başvurmanız gerekecektir. Bu kitapta 1541 disk sürücünün kullanımı adım adım açıklanmışsa da, verilen bilgileri tam olarak anlayabilmeniz için BASIC dilini ve disk işletimiyle ilgili komutları tanımanız şarttır. Ancak amacınız disk sürücünüzü hazır yazılım yüklemek ve saklamak için kullanmak ise, yalnızca bunun için ikinci ve üçüncü bölümü okumanız yeter.

Şimdi genel bilgiye devam edelim.

Disk sürücü için olan komutlar, karmaşıklık açısından birkaç seviyededirler. Üçüncü bölümden başlayarak, disk ile size programları saklamak ve yükleme imkanı veren komutların nasıl çalıştıklarını öğrenebilirsiniz. Dördüncü bölüm size, komutların diske gönderilmesini öğretecek ve disk bakım komutlarına bir giriş sağlayacaktır.

Beşinci bölüm size sırasal veri dosyalarıyla çalışmayı gösterecektir. Bunlar teypteki karşıtlarına benzer (fakat çok daha hızlıdır). Altıncı bölüm sizi rastgele dosyalarla çalışmayı, diskteki herhangi bir veri parçasına erişimi ve disketinizi bloklar ve izler şeklinde organize etmesini sağlayan komutlarla tanıştıracaktır. Yedinci bölüm, özel relatif dosyaları tarif etmektedir. Relatif dosyalar, özellikle sırasal dosyalarla birlikte kullanıldıkları zaman, veri tabanları depolamak için en iyi yöntemdir.

Sekizinci bölüm, disk kontrol devrelerinin makine dili seviyesinde programlanması için yöntemler açıklamaktadır. En son bölüm ise, sürücü ünitesinde donanım veya yazılım yolu ile cihaz numarasının değiştirilmesini anlatmaktadır.

Unutmayın ki, bu kitaptaki her şeyi aynı anda öğrenmeniz gerekmez. İlk dört bölüm, başlangıç için yeterlidir; sonraki ikisi ise çoğu işlemler için yeterlidir. Disket sürücünüzü tanımak birçok yönden çalışma hızı, güvenilirlik ve veri işleme kapasitesinde çok daha fazla esneklik, sizin için ödüllendirici olacaktır.

SPESİFİKASYON

Bu disk sürücü tek bir mini - flopi disket üzerinde, bilgi depolamaya uygun azami 174.000 byte için, 144 değişik program ve/veya veri dosyasına kadar depolama imkânı vermektedir.

Sürücüye dahil olarak, hem disk kontrolü hem de tüm bir disk işletim sistemi olmak üzere toplam 16K ROM ve 2K RAM bellek, bulunmaktadır. Bu devreler, Commodore 1541 disk sürücünüzü, ''zeki'' bir cihaz yapmaktadır. Bu demektir ki, Commodore 64 bilgisayarınızın belleğinden yer almaksızın, kendi işlemlerini kendisi yapmaktadır. Disk, ''Pipeline'' türü bir yazılım sistemi kullanmaktadır. ''Pipeline'' diskin, bilgisayar başka işler yaparken, komutları işleyebilmesini sağlar. Bu da sistemin input/output (giriş ve çıkış) yeteneklerini büyük çapta geliştirir.

Bu disk sürücüde yarattığınız diskler, Commodore 40-40 ve 2031 ve 1571 disk sürücülerce okuma ve yazmaya uygundur. Dolayısı ile disketler bu sistemlerin herhangi birinde kullanılabilir. Ayrıca bu sürücü, eski Commodore 2040 sürücülerde yaratılmış olan programları okuyabilir.

1541 disk sürücüde çiftli bir ''seri taşıyıcı'' arabirim bulunmaktadır. Bu taşıyıcı özel olarak Commodore tarafından yaratılmıştır. Bu taşıyıcının sinyalleri Commodore PET bilgisayarlarda kullanılan paralel IEEE-488 ara biriminkilere benzemekte, fakat veri iletişimi sekiz yerine bir tek tel üzerinden olmaktadır.

Sürücünün arkasındaki iki çıkış kapısı (port), aynı anda birden fazla cihazın seri taşıyıcıyı kullanabilmesini sağlar. Bu, cihazların ''zincirleme'' yöntemle, birbirine bağlanmaları ile sağlanmaktadır. Taşıyıcı aynı anda beş disk sürücü ve bir yazıcıya kadar cihaz paylaşabilmektedir.

Şekil 1.1 (Commodore 1541) Tek Flopi Disk Sürücü Özellikleri

DEPOLAMA	
Toplam kapasite	Disket başına 174848 byte
Sırasal	Disket başına 168656 byte
Relatif	Disket başına 167132 byte
	Dosya başına 65535-kayıt
Katalog giriş sayısı	Disket başına 144
Bant başına sektör sayısı	17 ila 21
Sektör başına byte sayısı	256
Bant sayısı	35
Blok sayısı	683 (Serbest 664)
ENTEGRE DEVRELER	
6502	Mikroprosesör
6522 (2)	I/O, iç zamanlayıcılar
2114 (4)	2K RAM Buffer
FİZİKİ	
Boyutlar	97 mm.
Yükseklik	200 mm.
En	374 mm.
Elektriksel:	
Güç ihtiyacı	220 VAC
Voltaj	50 Hertz
Frekans	25 Watt
Güç	

ORTAM:

Disketler	Tek taraflı, tek yoğunluklu standart mini 5 1/4'' (SSSD)

2. KUTUNUN AÇILMASI VE BAĞLANTILAR

KUTUNUN İÇİNDEKİLER

1541 Disk Sürücü birimi ile birlikte, gri bir cereyan kablosu, siyah seri taşıyıcı kablosu, bu el kitabını ve bir gösteri disketi bulacaksınız. Cereyan kablosunda, bir uçta disk sürücünün arkasına uygun bir bağlantı, diğer uçta ise topraklı (üç-uçlu) bir elektrik çıkışı vardır. Seri taşıyıcı kablosunun her iki ucu da aynıdır. 6-uçlu bir DIN fiş ile Commodore 64'e, C-16'ya, Plus/4'e veya diğer bir disk sürücüye veya bir yazıcıya bağlanır.

Lütfen bir sonraki bölümü okumadan hiçbir şey yapmayın!

Şekil 1: 1541'in iki ayrı modeli için ön panel şeması.

KABLOLARIN BAĞLANTILARI

İlk iş olarak cereyan kablosunu disk sürücünün arkasındaki AC girişine takın (şekil 2). Ters sokmaya çalışırsanız, girmeyecektir. Sürücüye taktıktan sonra, diğer ucu 220 Voltluk elektrik prizine sokun. Eğer disk sürücü bu sırada herhangi bir ses çıkarırsa, arkadaki cereyan anahtarını (Güç düğmesi) kapatın! Cereyan varken sürücüye başka hiçbir kablo takmayın!

Bundan sonra, seri taşıyıcı kablosunu alın ve sürücünün arkasındaki seri çıkış kapılarından birine takın. Bilgisayarı kapatın ve kablonun diğer ucunu bilgisayarın arkasına takın. Hepsi bu kadar!

Bir yazıcınız veya başka disk sürücüleriniz varsa, kablolarını ikinci seri çıkış kapısına takın (şekil 3). Aynı anda birden fazla sürücü kullanmak için, 9. bölümü okuyun. Eğer ilk defa bir sürücü kullanıyorsanız, birim ile rahat çalışabilene kadar tek bir sürücü kullanın.

ÇALIŞTIRMA

Tüm cihazlar bağlandıktan sonra, sıra çalıştırmaya gelecektir. Cihazların doğru sırada çalıştırılması önemlidir. Bilgisayar her zaman, en son çalıştırılmalıdır. En son çalışan bilgisayar olduğu sürece, her şey yolunda olacaktır.

Cereyan vermeden önce, sürücü içinde disket olmamasına dikkat edilmelidir.

Tüm diğer cihazlar çalıştırıldıktan sonra, bilgisayar çalıştırılabilir. Diğer tüm cihazlar ilk çalışma sıralarını izleyeceklerdir. Yazıcının motoru çalışacak ve başlık yarı yola kadar gidip geri gelecektir. 1541 Disk Sürücü'nün yeşil ışığı yanıyor olacak, kırmızı sürücü/hata ışığı ise, TV ekranında başlangıç resmi oluşurken, yanacak ve sönecektir.

Sürücü üzerindeki kırmızı sürücü/hata ışığının yanıp sönmesi bitince, sürücü kullanıma hazırdır.

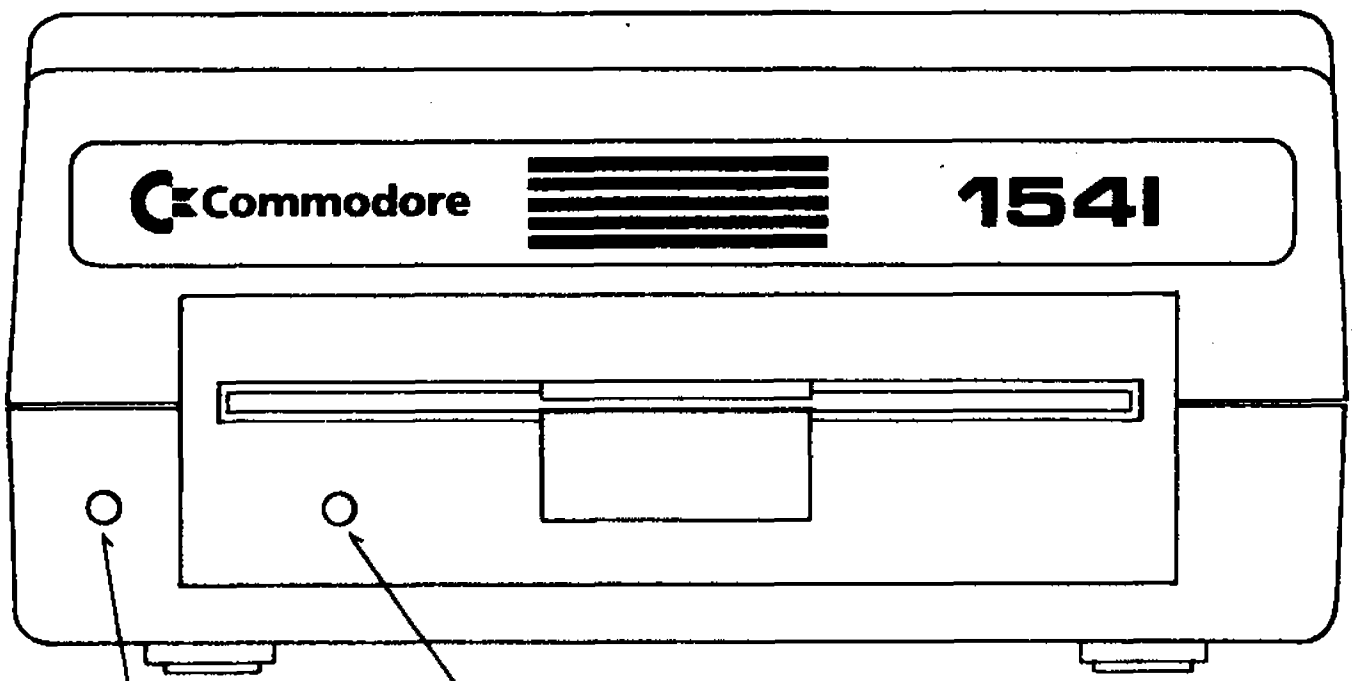
DİSKETİN TAKILMASI

Commodore 1541 Disk sürücüsünün iki ayrı modeli bulunmaktadır. Sizde bulunan cihazın hangi model olduğunu anlamak için **Şekil 1**'e bakın.

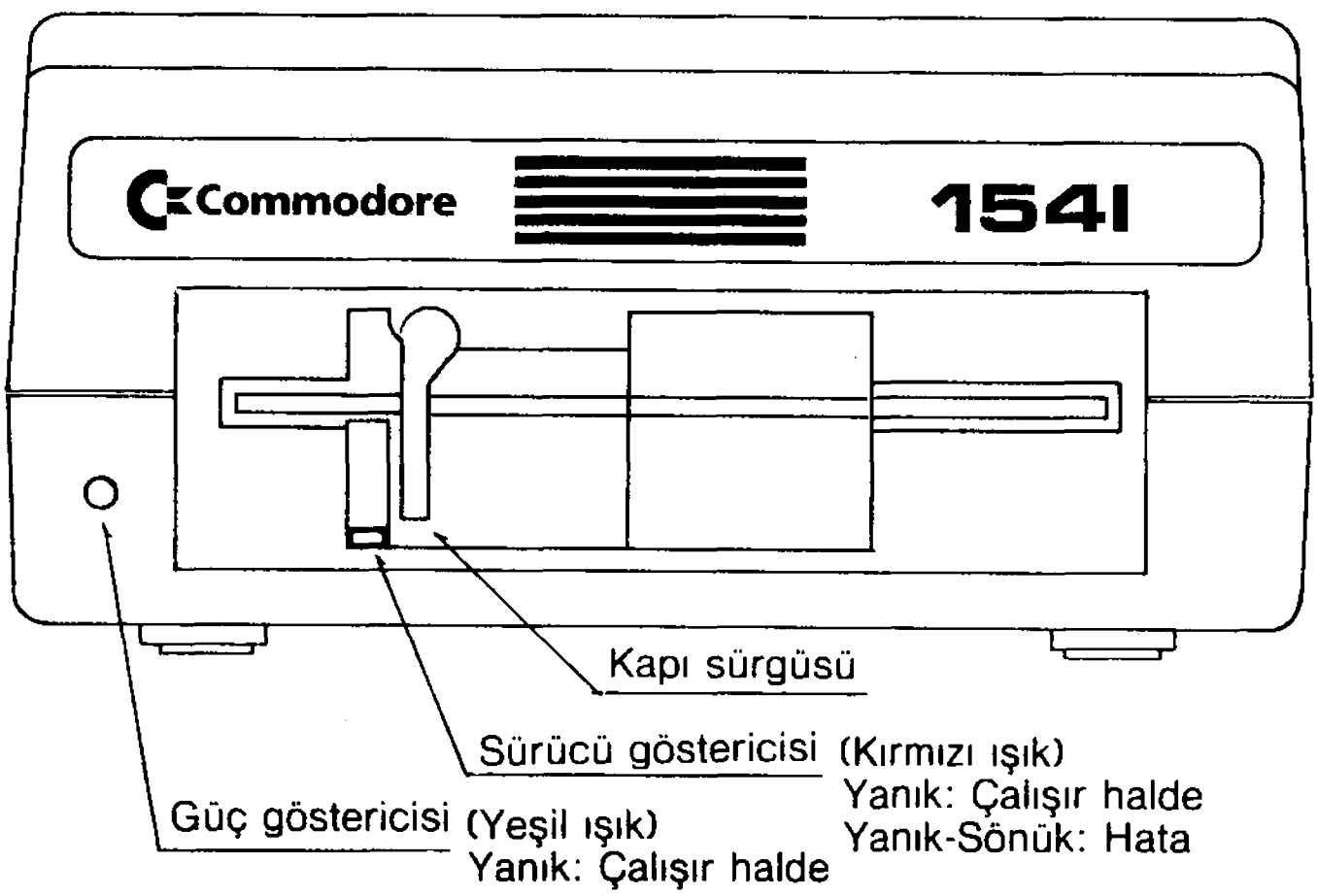
Kutunun İçindekiler

ŞEKİL 1
ÖN PANEL

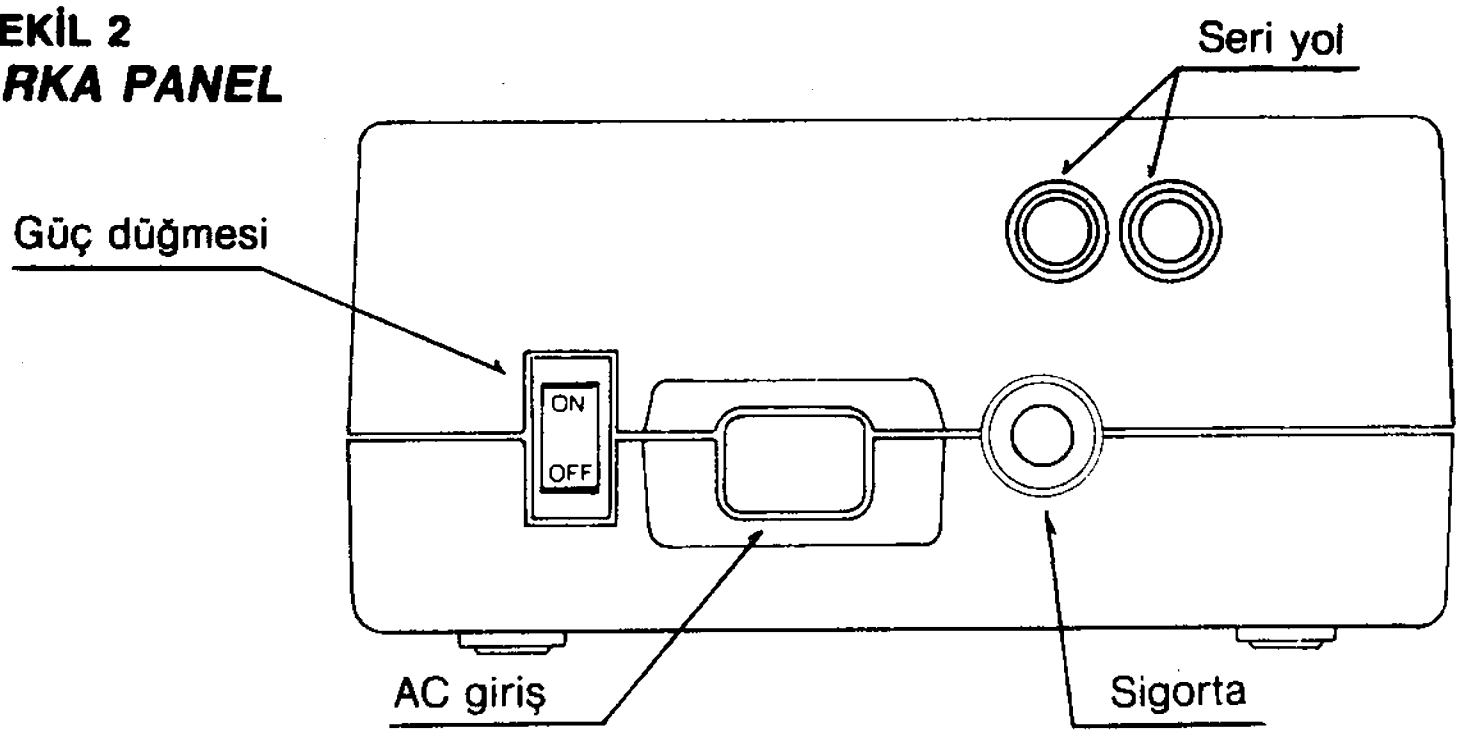
ŞEKİL 1 A



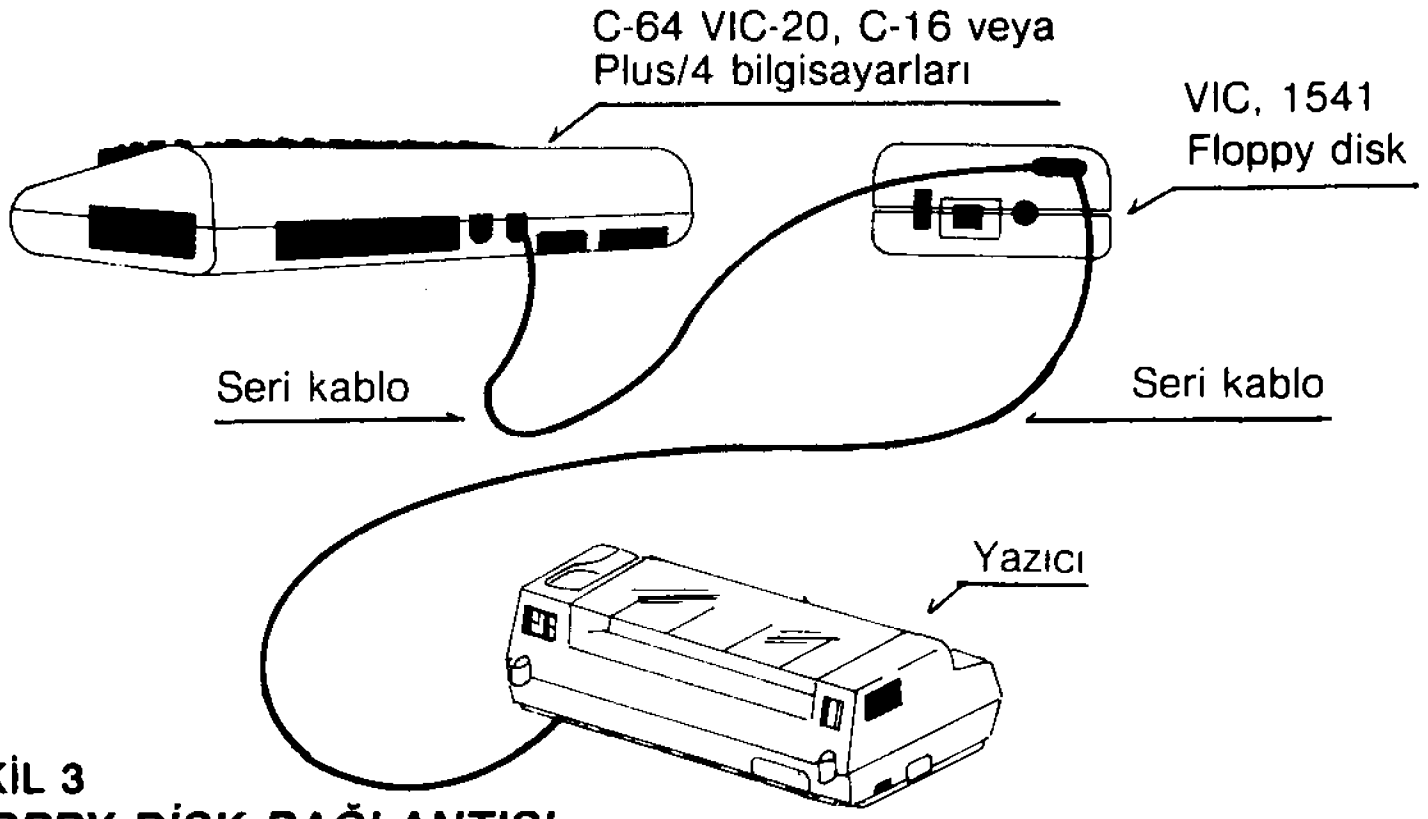
ŞEKİL 1 B



ŞEKİL 2
ARKA PANEL

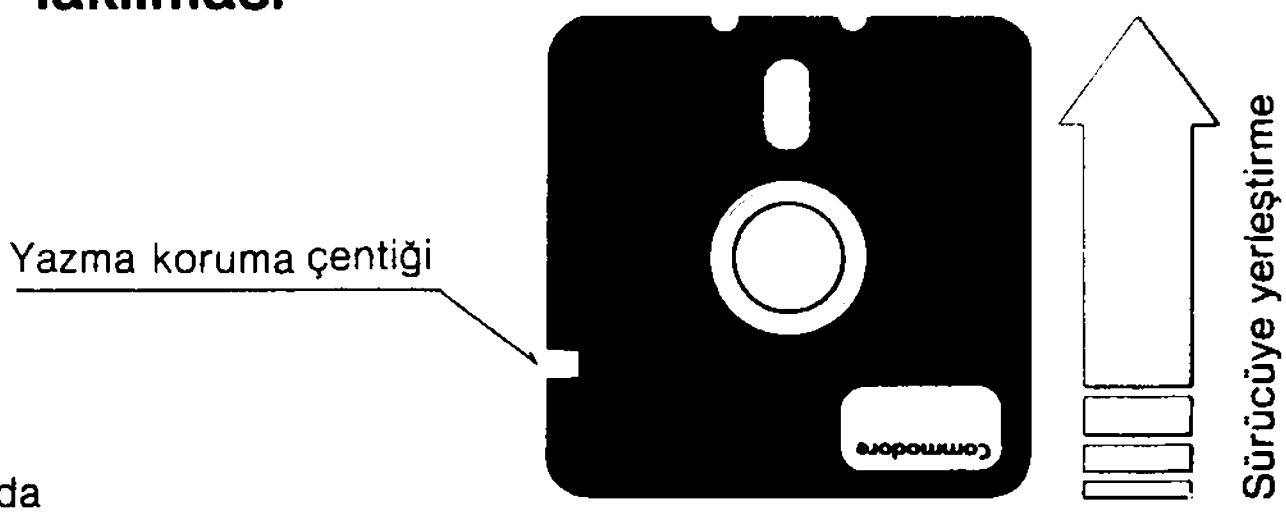


Kabloların Bağlantıları



ŞEKİL 3
FLOPPY DİSK BAĞLANTISI

Disketin Takılması



Kapandığında
Disket içeriği değişmez

MODEL 1: Şekil 1.A

Sürücü üzerindeki kapağı açmak için, kapak koluna hafifçe basın. Kapak açılacaktır. Eğer sürücüde bir disket varsa, küçük bir yay tarafından dışarı atılır. Takılacak disketi alın ve sürücüye yüzü yukarıya, yazım-koruma çentiği (gösteri diskinde bantlanmış) sola bakacak ve çıplak alan önce girecek şekilde takın (şekil 4).

Hafifçe içeriye bastırın. Disket tümüyle içeride iken bir çıtlama duyulur ve disket geriye gelmez. Sürücü kapağını, kilit yerine oturana kadar aşağıya bastırın. Artık disket ile çalışmaya hazırsınız.

MODEL 2: Şekil 1.B

Sürücü üzerindeki kapağı açmak için kapak mandalını saatin ters yönünde döndürün. Takılacak disketi alın ve sürücüye yüzü yukarıya, yazım-koruma çentiği (gösteri diskinde bantlanmış) sola bakacak ve çıplak alan önce girecek şekilde takın (şekil 4).

Mandalı, kilit çıtlayarak yerine oturana kadar, saat yönünde çevirerek kapağı kapatın. Artık diskle çalışmaya hazırsınız.

Sürücünün her kapatılışından veya açılışından önce disketi çıkarmayı unutmayın. Kırmızı sürücü ışığı yanarken hiçbir zaman disketi çıkarmayın! Bu esnada sürücü verileri bozabilir!

VIC 20 İLE KULLANIM

1541 Disk Sürücü VIC 20, Commodore 64, C-16 ve C-Plus/4 bilgisayarları ile çalışabilir. ancak her bilgisayarın veri alış hızı değişiktir. Dolayısı ile, hangi bilgisayarın hızı ile çalışacağı, bir yazılım anahtarı ile belirlenir. Sürücü normalde Commodore 64, C-16 ve Plus/4 için ayarlıdır. VIC 20 hızına geçmek için, sürücü çalıştırıldıktan (cereyan verilerek veya yazılım ile) sonra aşağıdaki komut verilmelidir:

```
OPEN 15,8,15, "UI—":CLOSE 15
```

Disk sürücüyü yeniden Commodore 64 hızına almak için ise, şu komutu kullanın:

```
OPEN 15,8,15, "UI+":CLOSE 15
```

Bu tip bir komutun kullanımı hakkında daha fazla bilgi bölüm 4'te, U (kullanıcı) komutları hakkında daha detaylı açıklama ise bölüm 8'de vardır.

DİSKETLERİNİZİ TANIYIN

Commodore 1541 disk sürücünüz 5 1/4 inch yarıçaplı standart bilgisayar disketleri kullanmaktadır. Tek yanlı tek yoğunluklu (SSSD) herhangi bir disketi 1541'de kullanabilirsiniz. Ancak, bir diskete kayıt yapabilmeniz için, disketin önce 1541'ce okunabilecek şekilde formatlanması gerekmektedir. Bu işlem için 4.cü bölümdeki NEW komutuna bakınız.

Disketiniz, manyetize edilmiş bir plastik daire ile koruyucu kılıfından oluşmaktadır. Kılıf üzerinde bir büyük bir de küçük pencere göreceksiniz. Büyük pencere, disket üzerindeki bilgilerin okunması içindir, küçük pencere ise disket dö-

nüş hızını ayarlayan sekronizasyon sinyalini okumakta kullanılır. Disket kılıfının kenarındaki çentik, koruma çentiğidir. Çentik kapatıldığı zaman disket okunabilir, fakat diskete kayıt yapılamaz. Disketi kayıt için kullanamıyorsanız, çentiği kapatarak yanlışlıkla programlarınızın bozulmasını engelleyebilirsiniz.

Disketlerinizi aşırı sıcak ve soğuğa, toz ve sıvılara karşı koruyun. Disket yüzeyine parmağınızı değdirmekten kaçının. Disketi, özellikle telefon, güçlü oparlör gibi manyetik alanlardan uzak tutun. Etiketin üzerine sert uçlu kalem veya tükenmezle yazmamaya dikkat edin.

Disketin kasetten önemli bir farkı, bildiğiniz gibi, okuma ve yazma hızının çok daha yüksek olmasıdır. Ama, gerçek bilgisayar uygulamaları için disketi kaçınılmaz kılan asıl önemli fark bu değildir. Disket, kasetten farklı olarak, rastgele bir kayıt ortamıdır.

Şöyle anlatalım: Kasetteki bilgiler baştan sona bir sıra halinde kaydedilir; mevcut bir programın veya kayıt için boş alanın bulunması için kasetin baştan sarılması gerekir.

Disket ise, 5 ve 6'ncı bölümlerde ayrıntılarıyla anlatılacağı gibi, içiçe halkalar halinde 35 bant ve toplam 683 bloka ayrılmıştır. Sürücü, bu bloklardan istenen herhangi birine, anında erişebilir. Disketin 18'inci bandında ise, disketin neresinde neyin bulunduğu kaydedilmiştir. Dolayısıyla, diskette bulunan program ve veri dosyalarının dökümü iki-üç saniye içinde çıkarılabilir; kaseti sarmak, yerini bulmak gibi kaygılar olmaksızın, istenilen program yüklenebilir, kaydedilebilir, veya değiştirilebilir.

3. PROGRAMLARI KULLANMA

HAZIR PROGRAMLARIN YÜKLENMESİ

Yalnızca diskteki hazır programları kullanmak isteyenler için, yapılacak şey şudur:

Disk sürücünüze, önceden programlanmış, diski, diskin üzerindeki etiket yukarıya bakacak ve size yakın olacak şekilde, dikkatlice takın. Eğer diski doğru takıyorsanız, çentik sol kenarda olacaktır. Disk içeri girince, koruyucu kapağı kapatın. Şimdi **LOAD "PROGRAMIN ADI", 8** yazın ve **"RETURN"** tuşuna basın. Diskten bazı sesler gelecek ve ekranınızda şu yazılacaktır:

SEARCHING FOR PROGRAMIN ADI
LOADING
READY

READY kelimesini görünce, **RUN** yazın; hazır yazılımınız kullanıma hazırdır. Bazı programlar, siz **RUN** yazmadan da çalışmaya başlayabilir.

LOAD

Disk sürücüdeki programlar için kullanılan BASIC komutları, Commodore DATASETTE teyp cihazında kullanılanların aynıdır. Ancak disklerle kullanılan birkaç fazladan komut vardır.

Öncelikle, her komut ile birlikte program adı verilmelidir. DATASETTE kullanıldığında teypteki ilk program yüklenmek istendiğinde, program ismi verilmeyebilir; ancak disk üzerinde aynı rahatlıkla erişilebilecek çok sayıda program olabileceği için, disk sürücüye ne yapacağını bildirmek için program adı veril-

melidir. Ayrıca, disket sürücünün cihaz numarası (8) belirtilmelidir. Cihaz numarası (8) belirtilmezse, bilgisayar, programın teypte olduğunu kabul edecektir.

LOAD KOMUTU İÇİN FORMAT:

LOAD isim$, cihaz no, komut no

Program adı bir dizidir; yani ya tırnak içinde bir isim ya da verilen bir dizi değişkenin içeriğidir. Bazı geçerli isimler: ''MERHABA'', ''PROGRAM 1'', A$, İSİM$

Cihaz no, devre üzerinde 8 olarak ayarlıdır. Birden fazla sürücünüz varsa, cihaz numarasını değiştirmek için bölüm 9'a bakın. Bu kitapta, disk sürücü cihaz numarasının 8 olduğu kabul edilmiştir.

Komut no, isteğe bağlıdır. Bu verilmezse, veya O ise, program normal yoldan LOAD edilir, yani yükleme BASIC'e açık belleğin başlangıcından başlar. Numara 1 ise, program geldiği bellek adresine LOAD edilir. Komut no BASIC programlarının normal yüklenmesini sağlar. Komut no 1 ise genellikle makina dili programlar, karakter dizileri ve diğer belleğe bağlı fonksiyonlar için kullanılır.

ÖRNEKLER

```
LOAD ''TEST'', 8
LOAD ''PROGRAM 1'', 8
LOAD A$, J,K
LOAD ''MAKİNA DİLİ PROGRAM'', 8, 1
```

NOT: Değişkenler, daha önce program içinde tanımlanmış oldukları sürece cihaz numaraları, komutlar ve dizileri temsil edebilirler.

DİSK KATALOĞU

DATASETTE, sırasal erişimli bir cihazdır. Yalnızca bantın başından sonuna kadar okuyabilir; bant parçalarını okumadan atlayamaz, geriye gidip eski verilerin üstüne kayıt yapamaz.

Disk sürücünüz bir rastgele erişim cihazıdır. Diskin okuma/yazma kafası disk üzerinde herhangi bir noktaya giderek, 256 byte bilgi taşıyan tek bir veri bloğuna erişebilir. Tek bir diskette toplam 683 blok vardır.

Veri blokları hakkında endişe etmenize gerek yoktur. Disk sürücü içinde Disk İşletim Sistemi, veya DOS, adında bir program vardır. Bu program blokları sizin yerinize kontrol eder, onları bir Blok Haritası (BAM) ve bir katalog şeklinde organize eder.

Blok Haritası, en basit şekliyle, disk üzerindeki 683 bloğun bir kontrol listesidir. Disketin ortasında, orta delik ve dış kenar arasında depolanır. Ne zaman bir program SAVE edilirse, veya bir veri dosyası CLOSE edilirse, BAM, kullanılan blokların bir listesini kapsayacak şekilde yenilenir.

Katalog, disk üzerindeki tüm program ve diğer dosyaların bir listesidir. Fiziksel olarak BAM'dan hemen sonra yer alır. Katalog içinde, dosya adı ve cinsi, kullanılan blokların listesi, ve bir başlangıç bloğu gibi bilgilerden oluşan, 144 giriş mevcuttur. Katalog, bir program SAVE edildiğinde veya bir dosya yazım için açıldığında yenilenir. DİKKAT: Dosya kapanmadan, katalog değişmekle beraber, BAM yenilenmez. Bir dosya doğru olarak kapatılmazsa, o dosyadaki tüm veriler muhtemelen kaybolacaktır.

Katalog, herhangi bir BASIC programı gibi belleğe yüklenebilir. Disketi sürücüye koyun, ve aşağıdaki komutu yazın.

LOAD "$", 8
Bilgisayar şöyle karşılık verecektir:
SEARCHING FOR $
FOUND $
LOADING
READY
Artık katalog, bilgisayarınızın belleğindedir. LIST yazın ve kataloğu ekranda izleyin. Kataloğu yazıcıya yazdırmak için, aşağıdaki komut satırını yazın (bu örnekte yazıcı, cihaz no 4 olarak bağlanmıştır).

OPEN 4,4: CMD 4:LIST

NOT: CMD kullanıldığında dosya, PRINT#4:CLOSE 4 komutu ile kapatılmalıdır. Detaylı açıklama için MPS 802 veya MPS 803 yazıcı el kitabına bakın. Kataloğu belleğe LOAD etmeden okumak için, bu bölümde yer alan, DOS destek programına ait kısmı okuyun. Ayrıca kataloğu bir BASIC programı içinden okumak için, GET # komutu ile ilgili olan kısmı 5. bölümden okuyun.

ÖRNEK EŞLEME VE JOKERLER

Teyp kullanılırken, belli harflerle başlayan bir programı, takip eden diğer harfleri yazmaksızın LOAD edebilirsiniz. Böylece, LOAD "T" komutu, Bantta T ile başlayan ilk programı bulacaktır. LOAD "SELAM" komutu, bantta SELAM harfleriyle başlayan ilk programı, örneğin "SELAMLAR" adlı programı bulacaktır.

Disk kullanıldığında bu opsiyona örnek eşleme denir, ve kullanılan dosya isminde özel bir karekterle belirtilir. Herhangi bir program ismini takip eden bir (*) karakteri sürücüye bu isimle başlayan herhangi bir program istediğini söyler.

ÖRNEK EŞLEME İÇİN FORMAT:

LOAD isim$*, 8 ←— isim$ bir dizi değişkeni veya tırnak içinde bir isim olabilir.

Başka bir deyişle, üzerinde T harfi ile başlayan ilk programı yüklemek istiyorsanız, LOAD "T*", 8 komutunu kullanın.

Eğer isim yerine yalnızca "*" kullanılmışsa, diskten katalogdaki ilk program yüklenir.

Muhtemelen pokerdeki joker kartlar kavramına aşinasınızdır. 1541'inizde soru işareti (?), disk üzerinde bir joker olarak kullanılabilir. Diskteki program ismi LOAD komutundaki program ismi ile karşılaştırılır, fakat soru işaretinin olduğu yerler kontrol edilmez.

Örneğin, LOAD "S?LAM", 8 komutu verildiğinde, uygun program isimleri SELAM, SALAM, SSLAM, vs. olabilir.

Disk kataloğu yüklenirken de örnek eşleme ve jokerler kullanılabilir. LOAD "$: TEST" komutunu vermişseniz (diskte mevcut ise) yalnızca TEST adlı program kataloğda listelenir. LOAD "$: T*" komutu size T harfi ile başlayan tüm programların kataloğunu, LOAD "$:T?ST" komutu ise tüm 4 harfli, birinci harfi T, üçüncü ve dördüncü harfleri ST olan programların kataloğunu, LOAD "$:ST*"

ise birinci ve ikinci harfleri ST olan her uzunluktaki tüm programların kataloğunu verecektir.

SAVE

Bir programı diskete SAVE (diskete kaydetmek) için gereken tek şey, program isminden sonra cihaz numarasını eklemektir. Teyp için kullanılan SAVE komutu gibi cihaz numarasını bir komut numarası takip edebilir ve yüklemeye otomatik dönüş engellenebilir (yukarıda LOAD komutu için olan kısmı okuyun).

SAVE KOMUTU İÇİN FORMAT

SAVE isim$, cihaz no, komut no

Cihaz no ve komut no parametreleri hakkında açıklama için LOAD komutuna bakınız.

Disk sürücüye bir programı SAVE etmesi söylendiği zaman, DOS'un bazı işler yapması gerekir. İlk olarak, bu isimde bir programın halen var olup olmadığını kataloğa bakarak öğrenir. Sonra bu isim için açık bir katalog girişi olup olmadığını kontrol eder. Daha sonra ise BAM'dan, programı depolamak için yeterli sayıda blok olup olmadığına bakar. Eğer buraya kadar herşey yolunda ise, program depolanır. Yoksa, hata ışığı yanıp söner.

SAVE VE DEĞİŞTİRME

Diskte halen bulunan programların sık sık, değiştirilerek yeniden SAVE edilmeleri gerekebilir. Bu durumda, eski programın silinerek yenisinin SAVE edilmesi gerekir.

DEĞİŞTİREREK SAVE İÇİN FORMAT

SAVE "@ø : isim$", cihaz no, komut no

Örneğin, eğer bir dosyanın adı TEST ise, değiştirerek SAVE komutu SAVE "@ø : TEST", 8 olacaktır. Eğer isim A$ ile temsil edilirse komut SAVE AS + "@ø : TEST", 8 olur. Ø: olmasının nedeni, birden fazla sürücülü Commodore disk sürücü birimleri ile uyuşmayı sağlamaktır. Bu durumda sürücüler Ø veya 1 olarak isimlendirilir. Tek sürücü kullanıyorsanız, değiştirerek SAVE için "@: İSİM" yeterlidir.

VERIFY

VERIFY komutu, halen bellekte olan programı, diskteki programla karşılaştırmak için kullanılır. VERIFY komutunda cihaz H belirtilmelidir. Bilgisayar, değişik bellek düzenlenmelerinde farklı olabilen, hat bağlantıları da dahil olarak programı byte-byte karşılaştırır.

VERIFY KOMUTU İÇİN FORMAT

VERIFY "isim$", cihaz no

DOS DESTEK PROGRAMI

Gösteri diskinizde, DOS SUPPORT adında bir program olabilir. "Wedge" (takoz) diye de anılan bu program birçok disk komutunu daha rahat kullanmanızı sağlar (VIC 20 ve Commodore 64 için farklı takozlar vardır). Programı LO-

AD ve RUN edin. Otomatik olarak kendini çalıştırır ve bittiği zaman kendini siler. Bu program çalışırken bellekten birkaç yüz bayt kaybedersiniz, fakat disk komutlarını göndermek için iyi bir yoldur.

DOS desteği sayesinde "l" tuşu LOAD komutunun yerini alır. Bölü işareti ve programın ismi yükleme için, yeterlidir. Bu yöntemi kullanırken, LOAD komutuna ve virgül ile 8'e ihtiyacınız yoktur.

"@" ve " > " tuşları sürücüye komut göndermek için kullanılır. Eğer @$ (veya $) yazarsanız katalog ekrana gelir ve belleğe LOAD edilmez! Bu anahtarlar aynı zamanda, bir sonraki bölümde listelenen komutlarda PRINT # (4. bölüme bakın)'ın yerini alırlar.

Diskin hata kanalını okumak için (kırmızı ışık yanıp sönerken) @ veya > anahtarı ile RETURN tuşuna basın. Tüm hata mesajı belirecektir: mesaj numarası, metin, bant ve blok numaraları.

4. DİSK KOMUTLARI

Disk komutları, bilgisayar tarafından disk sürücünün işletim sistemine gönderilen komutlardır. LOAD, SAVE gibi BASIC dilinde olan ve bilgisayar tarafından yerine getirilen komutların tersine, bu komutlar bilgisayar tarafından işlemez; tırnak içinde doğruca 1541'e iletilir.

Bilgisayarın disk komutlarını iletebilmesi için BASIC dilindeki OPEN ve PRINT# komutları kullanılır.

OPEN ve PRINT#

OPEN komutu, bilgisayara, başka bir cihazla konuşabilmesi için bir dosya ve bir kanal açmasını bildirir.

OPEN KOMUTUNUN FORMATI, dosya no, cihaz no, kanal no, (metin$)

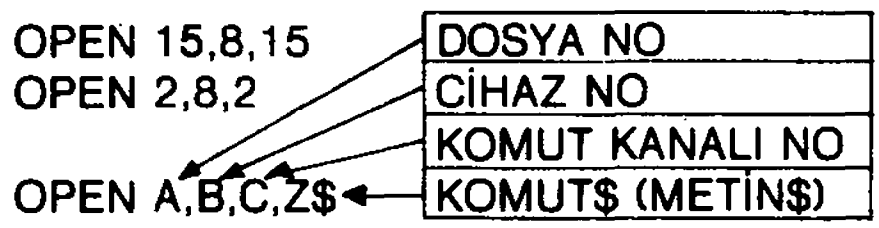
Dosya no, 1 ile 255 arasında herhangi bir sayı olabilir. Sayının önemi yoktur, fakat program boyunca, açılmış bir dosyaya ulaşmak için aynı sayıyı kullanmak gerekir. 127'den büyük rakamlardan kaçınılmalıdır, çünkü PRINT # komutunun her RETURN karakterinden sonra satır atlamasına neden olurlar. Bu sayılar standart olmayan yazıcılarda kullanım içindir.

Diskin cihaz no'su genellikle 8'dir.

Kanal no 2 ile 15 arasında herhangi bir sayıdır. Bunlar, diskle iletişimin kurulduğu kanalı belirtirler ve 0 ile 1 kanallar iletişim sistemine, LOAD ve SAVE yapabilmesi için, ayrılmıştır. 2. ile 14. kanallar dosyalara veri götürebilir, 15 ise komut kanalıdır.

Metin $, dosyaya, sonraki bir PRINT # komutu varmışçasına, yazılan bir dizidir. Bu da kanala tek bir komut göndermek için iyi bir yoldur.

OPEN KOMUTU ÖRNEKLERİ



PRINT # komutu tıpkı PRINT gibi çalışır. Yalnız, PRINT'in aksine veriler ekrana değil, komutu izleyen dosya no hangi cihaza açılmışsa o cihaza, örneğin disk sürücüye gönderilir.

PRINT # KOMUTU FORMATI:

PRINT #komutunu kullanabilmek için, verilen numarada bir dosyanın OPEN ile açılmış olması şarttır. Aksi halde FILE NOT OPEN hata mesajı alınır. Dikkat: Bu komut? şeklinde kısaltılamaz.

Disk Komutları Gönderme Formatı:

OPEN A, 8, 15 Komut $
veya
OPEN n, 8, 15
PRINT #n, Komut $

Şimdi disk komutlarını saymaya başlayabiliriz. Tekrar edelim: bu komutlar BASIC değildir, bilgisayarın işleyişiyle ilgili değildir, OPEN ve PRINT # yöntemiyle doğruca disk sürücüye iletilen mesajlardır.

NEW

Bir disketi ilk defa kullanmak için o disketi 1541'in yapısına göre formatlamak gerekir. Disket formatlamak için NEW komutu kullanılır. NEW komutu tüm disketi siler, disket üzerine zamanlama ve blok işaretleri koyar ve katalog ile BAM'ı oluşturur. NEW komutu aynı zamanda, formatlanmış bir disketin kataloğunu silmede kullanılabilir. Bu da, tüm diskin yeniden formatlanmasından daha hızlıdır.

DISK FORMATLAMADA NEW KOMUTU İÇİN FORMAT

PRINT # 15, "NEW: isim, tanım kodu" veya kısaca:
PRINT # 15, "N: isim; "tanım kodu"

İsim tüm diskin ismidir. Bu yalnızca katalog listelendiğinde belirir. Tanımlama kodu herhangi 2 karakterdir ve bunlar yalnızca kataloğa değil disket üzerinde her bloğa yazılırlar. Böylece veri yazarken yanlışlıkla disket değiştirirseniz, sürücü tanımlama kodundan hatayı anlayacaktır.

Dikkat: NEW ile formatlanan disketteki tüm bilgiler tamamen silinecektir.

COPY

Bu komut size disk üzerindeki herhangi bir program dosyasının bir kopyesini çıkartma imkanı verir. Bu sürcüden diğerine kopye yapmamakla birlikte, bir programı sürücüde başka bir isim altında kopye edebilir.

COPY KOMUTU İÇİN FORMAT:

PRINT # 15, "COPY: yeni dosya = eski dosya"
vaya kısaca:
PRINT # 15, "C: yeni dosya = eski dosya"

COPY komutu aynı zamanda iki ila dört dosyayı birleştirmede kullanılabilir.

DOSYA BİRLEŞTİRMEDE COPY İÇİN FORMAT

PRINT # 15, "C: yeni dosya = eski dosya 1, eski d. 2, eski d. 3, eski d. 4"

COPY KOMUT ÖRNEKLERİ

PRINT # 15, "C: YENİ = ESKİ"
PRINT # 15, "C: TOPLU DOSYA = İSİM LİST, ADRESLIST, TELLIST"

RENAME

Bu komutla, disk üzerinde olan bir dosyanın ismi değiştirilebilir. Yalnızca kataloğdaki isim değiştiği için, bu hızlı bir süreçtir.

RENAME KOMUTU İÇİN FORMAT:

PRINT # 15, RENAME: yeni ad = eski ad" veya kısaca
PRINT # 15, "R:yeni ad = eski ad"

RENAME KOMUT ÖRNEKLERİ

PRINT # 15, "R:HASAN = HÜSEYİN"
RENAME komutu, halen açık olan dosyalarda çalışmaz.

SCRATCH

Bu komut, yeni bilgiler için serbest blok ayırmak için, diskten istenmeyen program ve dosyaların silinmesini sağlar. Programları birer birer veya, örnek eşleme ve/veya joker kullanarak, gruplar halinde silebilirsiniz.

SCRATCH KOMUTU İÇİN FORMAT:

PRINT # 15, "SCRATCH: isim, isim, isim"
veya kısaca
PRINT # 15, "S: isim"

Eğer bir silme işleminden sonra hata kanalını kontrol ederseniz (aşağıda), genellikle bant numarası için ayrılan rakam size silinen dosya sayısını verir. Örneğin, eğer katalogda SELAM ve SALAM programları varsa ve komutunuz PRINT #15; "S: S?LAM" ise, her iki programı da silersiniz. Katalogda TEST, TAHMİN, TALAT1, TALAT2 varsa ve diskte PRINT # 15, "S:T*" komutunu verirseniz, dört rogram da silinir.

INITIALIZE

Bazen disk üzerindeki bir hata durumu belli bir işlemi yapmanızı engelleyebilir. INITIALIZE komutu disk sürücüyü ilk açıldığı zamanki durumuna getirir. VIC-20'de sürücüyü yeniden bilgisayara ayarlamanız gerekebilir. (Bölüm 2)

INITIALIZE KOMUTU İÇİN FORMAT:

PRINT #15, "INITIALIZE"
veya kısaca:
PRINT # 15, "I"

VALIDATE

Bir disket bir süre kullanıldıktan sonra katalog düzeni bozulabilir. Program sürekli SAVE ve SCRATCH edildikçe disk üzerinde, orada burada bir veya birkaç blokluk küçük boşluklar bırakabilirler. Bu bloklar hiç kullanılmaz çünkü çok ufaktırlar. VALIDATE komutu disketinizi, mevcut boş alandan en iyi şekilde ya-

rarlanmanızı sağlamak için yeniden düzenler.

Ayrıca, açılmış fakat iyi kapatılmamış veri dosyaları da olabilir. Bu komut, böyle dosyalar tarafından tutulan tüm blokları sürücüye serbest hale getirir, çünkü dosyalar artık kullanılamaz durumdadır.

Bu komutun kullanımında bir tehlike vardır. Rastgele dosyalar kullanıldığında (bölüm 6), açılan bloklar bu komut tarafından geri alınır. Dolayısıyla, bu komut rastgele dosyaların bulunduğu bir disketle hiçbir zaman kullanılmamalıdır.

VALIDATE KOMUTU İÇİN FORMAT:

PRINT # 15, "VALIDATE"
veya kısaca:
PRINT # 15, "V"

HATA KANALININ OKUNMASI

Diskete kayıt veya disketten okuma işlemleri sırasında bir hata yapıldığında, sürücünün ön tarafındaki kırmızı ışık yanıp sönmeye başlar. Hata sürücüde olduğu için, hata mesajı otomatik olarak ekrana gelmez; bilgisayara, sürücüden, hata türünü sorması bildirilmelidir.

DOS destek programı olmaksızın, programsız disk hata kanalının okunması mümkün değildir, çünkü INPUT # komutuna ihtiyaç vardır, bu da program dışı çalışmamaktadır. Aşağıda, hata kanalını okumak için basit bir BASIC rutin programı vardır:

```
10 OPEN 15, 8, 15
20 INPUT # 15, A$, B$, C$, D$
30 PRINT # A$, B$, C$, D$
```

Komut kanalından her bir INPUT # işlemi yapıldığında, hata durumunu tanımlayan 4 değişken okunur. İlk, üçüncü ve dördüncü değişkenler rakamdır, ve istendiğinde nümerik değişkenler olarak INPUT edilebilirler. İlk rakam hata tanımlar; O, hata yok demektir. İkinci değişken hatanın tanımıdır. Üçüncü değişken hatanın bulunduğu bant sayısı, dördüncü ve son değişken ise o bant içindeki blok içindeki blok numarasıdır (Blok, aynı zamanda sektör olarak da adlandırılır.) Disk sürücüden gelebilecek tüm hata mesajlarının listesi ek B'de verilmiştir.

18'inci banttaki hatalar BAM ve katalogla ilgilidir. Örneğin, bant 18 blok O'da bir READ ERROR, diskin hiç formatlanmadığını gösterir.

CLOSE

Kullanımı biten dosyaların kapatılması çok önemlidir. Dosyanın kapatılması, DOS'un BAM'de blok ayırmayı ve kataloga girişi bitirmesini sağlar. Dosyayı CLOSE etmezseniz, tüm verileriniz kayıp olacaktır.

CLOSE KOMUTU İÇİN FORMAT:

CLOSE dosya no

Ayrıca, veri kanalları kapatılmadan, hata kanalı (kanal 15) kapatılmamalıdır. Tüm dosyalar içinde hata kanalı ilk OPEN edilen ve son CLOSE edilen olmalıdır. Bu programlarınızı problemli olmaktan korur.

Diğer dosyalar açıkken hata kanalını kapatırsanız, disk sürücü diğer dosyaları kapatır, fakat BASIC bunların hâlâ açık olduğunu zanneder ve bunlara yazmanıza izin verir.

NOT: Eğer BASIC programınızda hata olursa, disk sürücüde açık kalmakla birlikte, tüm dosyalar BASIC'te kapanır! Bu çok tehlikeli bir durumdur. Derhal OPEN 15, 8, 15, "I" komutunu yazmalısınz. Bu sürcünüzü başlangıç durumuna döndürerek tüm dosyalarınızı koruyacaktır.

5. SIRASAL DOSYALAR

15431 Disk sürücü, diskte dört ayrı çeşit dosya açma ve okuma imkânına sahiptir: bunlar, program, Sequential (sırasal), relatif ve user (kullanıcı tarafından tanımlanan) dosyalardır. Program dosyaları SAVE ile açılır. LOAD ile okunur; BASIC veya makina dili programların saklanmasında kullanılır. Sırasal ve relatif dosyalar, bir programda oluşturulmuş verilerin (örneğin adres listeleri, sayı dizileri) saklamasına yarar. Sırasal dosyalama yöntemi, veri saklamanın basit türüdür; bir bakıma kaset gibi -ama çok daha hızlı- işler; verileri ucuca dizilmiş bir sıra halinde saklar.

OPEN

Sırasal dosyalar LOAD ve SAVE edilmez; OPEN ve PRINT # komutlarıyla açılır ve OPEN ve GET # veya INPUT # komutlarıyla okunur. OPEN ile açılan her dosyanın, işi bittiğinde CLOSE ile kapatılması şarttır.

SIRASAL DOSYA AÇMAK İÇİN FORMAT:

OPEN dosya no, cihaz no, kanal no, "O: isim, tür, işlem"

Dosya no, OPEN komutunun diğer örneklerinde olduğu gibidir ve program boyunca bu dosyayı tanımlamaya yarar. Cihaz no genellikle 8'dir. Kanal no, 2-14 arasında bir veri kanalıdır. Kanal ve dosya numaralarını aynı tutmakta yarar vardır. İsim, dosyanın ismidir (Yazım için dosya açıyorsanız joker ve en azından tipin baş harfi kullanılmalıdır. İşlem READ (OKU) veya WRITE (YAZ), ya da bunlardan birinin baş harfi olabilir.

DOSYA TİPİ	ANLAMI
PRG	Program
SEO	Sırasal
USR	Kullanıcı
REL	Relatif

SIRASAL DOSYA AÇMA ÖRNEKLERİ:

```
OPEN 2, 8, 2, "DATA, S, W"
OPEN 8, 8, 8, "PROGRAM, P, R"
OPEN A, B, C, "A$ + "U, W"
```

Eğer dosya halen varsa, OPEN komutunda, bölüm 3'de söz edilen SAVE ve değiştirmedeki gibi, değiştirme opsiyonunu kullanabilirsiniz.

DEĞİŞTİRMELİ SIRASAL DOSYA ÖRNEĞİ:

OPEN, 2, 8, 2, ''@O: DATA, S, W''

Eğer dosyanın isminden önce O: olmazsa sürücü size mevcut tamponlardan yalnızca 2 tane ayıracaktır.

PRINT # VE INPUT

Çıkışın yeniden sürücüye verilmesi dışında, PRINT # komutu PRINT komutu ile aynı çalışır. Aynı kelimesindeki özel vurgunun anlamı, PRINT komutunun, noktalama ve veri tiplerine dair, tüm formatlama yeteneklerinin burada da geçerli olduğudur. Bunun anlamı ise, dosyalarınıza verilerinizi yerleştirirken dikkatli olmamız gerektiğidir.

PRINT # İLE DOSYAYA YAZMAK İÇİN FORMAT:

PRINT # dosya no, veri listesi

Dosya no, dosya yaratılırkenki OPEN komutunda yazılı olandır.

Veri listesi normal PRINT komutunda olduğu gibidir. Değişken listesi ve/veya tırnak içinde metinler. Ancak veri yazarken, daha sonra okumanın en kolay olacağı biçimde yazmaya özen göstermelisiniz.

PRINT # komutu ile, satırdaki elemanları ayırmak için virgül (,) kullanırsanız, elemanlar aynen ekrandaki gibi, başlıklarla ayrılacaklardır. Noktalı virgüller (;), fazladan boşluğa neden olmazlar.

Ne olduğunu daha iyi anlamak için, OPEN5, 8, 5 ''O:TEST, S, W'' komutuyla yaratılan aşağıdaki sırasal dosya diyagramına bakalım:

	eof															
karkt.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16...

Eof, dosya - sonu işaretidir. Dosyaya giren dizi veriler, boşluklar dahil, byte-byte girerler.

Örneğin, şu değişkenleri yaratalım: A$ = ''SELAM''; B$ = ''İYİ''; C$ = ''GÜNLER''. PRINT # 5, A$; B$; C$ komutundan sonra dosyanın durumu şöyledir.

	S	E	L	A	M	İ	Y	İ	G	Ü	N	L	E	R	CR	eof
karkt.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16...

CR, 13'ün CHR$ kodu, yani satırbaşı işaretidir; satırın sonunda virgül veya noktalı virgül olmadıkça, her PRINT veya PRINT # komutundan sonra PRINT edilir.

NOT: PRINT ve # arasında boşluk bırakmayın ve komutu ? şeklinde kısaltmayın. Doğru kısaltma için kullanıcı el kitabındaki eklere bakın.

INPUT # KOMUTU İÇİN FORMAT

INPUT # dosya no, değişken listesi

INPUT # ile veri okurken, sistem okuduğunun tek bir dizi olup olmadığını anlayamaz. Dosyada, ayırıcı olarak bir şeye ihtiyaç vardır. Ayırıcı olarak kullanı-

lan karakterler CR, virgül veya noktalı virgüldür. PRINT komutuna yalnızca bir değişken eklendiğinde sistem, otomatik olarak, bir sonrakinden önce CR'ı ekleyecektir. PRINT # 5, A$: PRINT # 5, B$: PRINT #5, C$ komutu, INPUT#5, A$, B$, C$ gibi bir komut için uygun bir ayırım sağlayacak şekilde her değişkenden sonra bir CR ekler. Ya da Z$ = '','': PRINT # 5, A$ Z$ B$ Z$ C$ komutu da, hem de daha az yer kaplayarak, aynı işi görür. Bu satırdan sonra dosya şu hale gelir:

	S	E	L	A	M	,	İ	Y	İ	,	G	Ü	N	L	E	R	CR	eof
karkt.	1	2	3	4	5	6	7	8	9	10	12	13	14	15	16	17		

Değişkenler arasında virgül konması diskte fazladan yer kullanımına neden olur. PRINT # 5, A$ B$, gibi bir komut dosyayı şu hale getirir.

	S	E	L	A	M							İ	Y	İ	CR	eof
karkt.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...23	24

Görüldüğü gibi dosyadaki yerin çoğu ziyan olmaktadır.

Bunların anlamı şudur: PRINT # kullanırken, verilerinizin düzenli bir şekilde okunabilmesi için, dikkatli olun.

Dosyaya yazılan nümerik veriler, yazılmadan önce sanki STR$ fonksiyonunu kullanılmış gibi, dizi şeklini alır. Rakam pozitifse ilk karakter boşluk, rakam negatifse ilk karekter eksi (-) olacaktır. Sonra rakam gelmektedir. Son karakter ise sağa yön gösterici karakterdir. Bu format, özel bir ayırıcı olmadan birkaç tane yazılmışsa, INPUT # komutuna bunları değişik rakamlar olarak okuma imkanı verir.

Dosyanın PRINT # 5, 1;3;5;7 komutunun yerine getirilmesinden sonraki durumu aşağıdadır.

		1	→		3	→		5	→		7	→	CR	eof	
karkt.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

EK C'de, sırasal bir disk dosyasının kullanımını gösteren bir program vardır.

GET #

GET # diskten, her defada bir karakter olarak veri alır.

GET # KOMUTU İÇİN FORMAT:

Get # dosya no, değişken listesi

Veriler, CR virgül ve diğer ayırıcı karakterler dahil, byte-byte alınır. GET # komutu ile, dizisel değişken kullanmak çok daha emindir. Rakam istendiği yerde dizi veri alınırsa bir BASIC hata mesajı verilmekle beraber, bunun tersi sözkonusu değildir.

GET # KOMUT ÖRNEKLERİ:

```
GET # 5, A$
GET # A, B$,C$,D$
GET # 5, A
```

Bir kerede birden fazla karakter alabilirsiniz.

GET # komutu, bir deneme programı tarafından bir veri dosyası gibi, içeriği bilinmeyen dosyaların incelenmesinde son derece yararlıdır. INPUT # dan daha emindir, çünkü INPUT değişkenlerinde, ayırıcılar arasındaki karakter sayısı kısıtlıdır. GET # ile tüm karakterleri alabilir ve ayırıcıları da diğer karakterler gibi inceleyebilirsiniz.

Şöyle bir programla, disk üzerindeki herhangi bir dosyayı inceleyebilirsiniz.

```
10  INPUT ''DOSYA İSMİ''; F$
20  INPUT ''DOSYA TİPİ''; T$
30  T$ = LEFT$ (T$,1)
40  IF T$ <>''S'' THEN IF T$ <> ''P'' THEN IF T$ <>''U'' THEN 20
45  OPEN 15, 8, 15
50  OPEN, 5, 8, 5, ''O:'' + F$ '','' + T$ +, ''R''
60  GOSUB 200
70  GET # 5, A$
80  IF ST=0 THEN 90
85  IF ST=64 THEN CLOSE5, 15: END
86  PRINT ST: STOP
90  PRINT ASC (A$ + CHR$ (0)):
100 GOTO 70
200 INPUT # 15, A$, B$, C$, D$
210 IF VAL (A$) >O THEN PRINT A$, B$ C$ D$: STOP
220 RETURN
```

KATALOĞUN OKUNMASI

Disketin kataloğu herhangi bir sırasal dosya gibi okunabilir. Dosya ismi olarak $ kullanın ve OPEN 5, 8, 5, ''$'' komutunu verin. GET # komutu, kataloğu incelemeye yarar. Buradaki format bir program dosyasının formatının aynıdır. Dosya büyükleri satır numaraları, isimler ise tırnak içinde karakterlerdir.

İşte disketin kataloğunu okumak için bir program:

```
10 OPEN 1, 8, 2, ''$''
20 FOR X=1 TO 141:GET 1, A$:NEXT
30 T$ (O)=''DEL'';T$ (1)=''SEQ'': T$ (2)=''PRG'':T$ (3)= ''USR'': T$
(4)=''REL''
40J=17: GOSUB 500
50N$=B$
60 J=2
70 GOSUB 500
80 I$=B$
85 GET # 1, A$
```

BAM'yi geç

Disk ismi

Tanım kodu dizisinin uzunluğu

```
90 J=2 ←— İşletim sistemi dizisinin uzunluğu
100 GOSUB 500
110 O$=B$
120 FOR L=1 TO 88
130 GET #1, A$ ←— Bloktan geri kalan
140 NEXT
160 PRINT CHR$ (147) ''DİSK ADI:'' N$, ''ID:'' 1$, ''SİSTEM:'' O$
161 PRINT ''UZUNLUK'', ''TİP'', ''İSİM''
165 FOR P=1 TO 8
170 GET #1, T$, A$, A$          Dosya tipi
180 IF T$='' '' THEN T$=CHR$ (128)
190 J=15 ←— Dosya ismi dizisinin uzunluğu
200 GOSUB 500
210 N$=B$
220 GET# 1, A$, A$, A$, A$, A$, A$, A$, A$, A$, A$, L$, H$ ←
225 L=ASC (L$+CHR$(O))+256*ASC (H$+CHR$(O)) IF L=O THEN 260
227 IF ST THEN CLOSE 1: END
230 PRINT L, T$ (ASC (T$) - 128), N$
250 IF P < 8 THEN GET # 1, A$, A$
260 NEXT P: GOTO 165
500 B$='' ''
510 FOR L=O TO 5
520 GET# 1, A$
530 IF A$< >CHR$ (96) THEN IF A$ < >CHR$ (160) THEN B$=B$ + A$
540 NEXT
550 RETURN
```

Dosya uzunluğunun alt ve üst sınır baytları

TABLO 5.1. 1650/1541 BAM FORMATI		
BANT 18, SEKTÖR 0		
BYTE	**İÇERİĞİ**	**TANIM**
0,1	18,01	İlk katalog bloğunun bant ve sektörü
2	65	4040 formatının ASCII A karakteri
3	0	
4-143		1-35 bantlar için boş blokların bit haritası

*1— Boş blok
0- Dolu blok
(Her bit bir bloğu temsil eder)

TABLO 5.2. 1540/1541 KATALOG BAŞLIĞI		
BANT 18, SEKTÖR 0		
BYTE	**İÇERİĞİ**	**TANIM**
144-161		Disk ismi
162-163		Disk tanım kodu
164	160	SHIFT SPACE
165-166	50,65	DOS versiyonu ve format tipi olan, 2A'nın ASCII İLE TEMSİLİ
166-167	160	SHIFT SPACE
171-255	0	Boş
Not: Bazı disketlerde ASCII 2A karakterleri 180-181 arasında olabilirler.		

TABLO 5.2 KATALOG FORMATI

BANT 18, SEKTÖR 1	
BYTE	**TANIM**
0,1	Bir sonraki katalog bloğunun bant ve sektörü
2-31	* Madde 1
34-63	* Madde 2
66-95	* Madde 3
98-127	* Madde 4
130-159	* Madde 5
162-191	* Madde 6
194-223	* Madde 7
226-255	* Madde 8

TEK BİR KATALOG MADDESİNİN YAPISI

BYTE	İÇERİĞİ	TANIM
0	128+TİP	Dosya tipi, normal kapanış dosya göstergesi olarak,
		TİPLER: 128—DEL (boş) 129—SEQ (sırasal) 130—PROG (program) 131—USER (kullanıcı) 132—REL (relatif)
1,2		İlk veri bloğunun bant ve sektörü
3-18		Dosya adı
19,20		Yalnızca relatif dosya: ilk kenar sektör bloğunun bant ve sektörü
21		Yalnızca relatif dosya: Kayıt büyüklüğü
22-25		Kullanılmaz
26,27		OPEN C çalışırken, değiştirilen dosyanın bant ve sektörü
28,29		Dosyadaki blok sayısı: alt byte, üst byte

TABLO 5.4. SIRASAL FORMAT

BAYT	TANIM
0,1	Bir sonraki sırasal veri bloğunun bant ve sektörü.
2-256	Satır atlamaların kayıt uçları olduğu, 254 veri baytı.

TABLO 5.5. PROGRAM DOSYA FORMATI

BAYT	TANIM
0,1	Program dosyasında bir sonraki bloğun bant ve sektörü.
2-256	CBM bellek formatında depolanan 254 byte'lık program bilgisi Dosya sonu, üç tane sıfır byte ile işaretlenir.

6. RASTGELE ERİŞİM

Sırasal dosyalar, sürekli bir veri akımı ile çalışırken iyidir, ama bazı işler bundan fazlasını gerektirir. Örneğin, bir adres rehberinde bir kişinin kaydını bulmak için tüm listeyi taramak istemezdiniz. Bunun için, bir şekilde tüm listeyi taramadan belli bir kaydı bulmak için, bir rastgele erişim metodu gereklidir.

Commodore disk sürücüde iki rastgele erişim dosyası yöntemi vardır. Bu bölümde anlatılan rastgele erişim yöntemi, özellikle makine dili ile çalışılırken kendine özgü kullanımları olmakla birlikte, verilerle uğraşmada bir sonraki bölümde söz edilen relatif dosyalar daha rahattır.

Bu bölümde anlatacağımız yöntemi anlamak için C-1541 DOS'unun (Disk İşletim Sisteminin) diski nasıl bölümlediğini bilmek gerekiyor. Daha önce de değindiğimiz gibi, yeni bir disketi formatladığımızda, disket, içiçe halkalar şeklinde 35 banta ayrılacaktır. Her bant ise, kendi içinde, 256'şar byte bilgi içerebilen sektörlere (bloklara) bölünür. Her bant'ın sektör sayısı farklıdır. Dış bantlar, daha uzun oldukları için 21 sektöre ayrılır. İç bantlar ise daha kısadır.

Aşağıdaki tablo, bant başına blok sayısını göstermektedir.

TABLO 6.1 BANT VE BLOK FORMATI

Bant No	Bloklar	Toplam Blok
1-17	0-20	21
18-24	0-18	19
25-30	0-17	18
31-35	0-15	16

DOS, disket üstünde numarası verilen herhangi bir bant ve bloğa yazmak üzere komutlara sahiptir. Ayrıca, boş blokları aramak ve dolu blokları işaretlemek için de komutlar vardır.

Bu komutlar, komut kanalından (kanal 15) aktarılır ve diske verilerine yapacağını bildirirler. Sürücüden gelen veriler daha sonra, açık bir veri kanalından (kanal 2-14) okunabilir.

RASTGELE ERİŞİM İÇİN BİR KANAL AÇILMASI

Rastgele erişim dosyalarıyla çalışırken, birisi komutlar, diğeri veriler için olarak 2 kanalı diske açmanız gerekir. Komut kanalı, her zaman olduğu gibi kanal 15'dir. Rastgele dosyalar için veri kanalı, dosya ismi olarak '' '' yazılarak açılır.

RASTGELE ERİŞİM VERİLERİ İÇİN OPEN KOMUT FORMATI

OPEN dosya no, cihaz no, kanal no, ''#'' veya
OPEN dosya no, cihaz no, kanal no, ''# tampon no''.

RASTGELE ERİŞİM VERİ KANALI AÇMA ÖRNEKLERİ

OPEN 5,8,5,''#'' — Tampon önemli değil

OPEN A,B,C, ''#2'' — 2'nci tampon seçim

BLOCK-READ

BLOCK-READ KOMUT FORMATI

PRINT # dosya no, ''BLOCK-READ: ''kanal; sürücü; bant; blok veya kısaca:
PRINT # dosya no, ''B-R:'' kanal; sürücü; bant; blok
Bu komut disketten bir blok veriyi seçilen kanala alır. Bu işlem bitince, INPUT # ve GET # komutları ile bilgi okunabilir.

Bant 18, Blok 2'yi DEN OKUYAN ÖRNEK PROGRAM

Kanal — Sürücü — Bant — Blok

```
10 OPEN 15, 8, 15
20 OPEN 5, 8,5 ''#''
30 PRINT # 15, ''B-R:'' 5; 0; 18, 2
40 B$ = ''''
50 FOR L = 0 TO 255
60 GET# 5,A$
70 IF ST = 0 THEN B$ = B$ + A$: NEXT L
80 PRINT ''Bitti''
90 CLOSE 5: CLOSE 15
```

Tüm bloğu, byte-byte topla

BLOCK-WRITE

BLOCK-WRITE komutu BLOCK-READ komutunun tam tersidir. Bu veri tamponu önce bilgi ile doldurulur, sonra bu tampon disk üzerinde doğru yere yazılır.

BLOCK-WRITE KOMUT FORMATI

PRINT # dosya no, "BLOCK-WRITE: "kanal; sürücü; bant; blok veya kısaca:
PRINT # dosya no, "B-W:" kanal; sürücü; bant; blok.

Veriler tampona konurken, DOS'taki bir işaret edici, kaç karakter olduğunu hesaplar. BLOCK-WRITE işlemi yapılırken diskte bu kayda gerek vardır. Bu, yukarıdaki programda 70 no'lu satırdaki ST kontrolünün nedenidir. Kayıt içinde dosya-sonu işaretinin ötesine geçildiğinde ST, sıfırdan büyük olacaktır.

BANT 1, SEKTÖR 1'E VERİ YAZMAK İÇİN ÖRNEK PROGRAM:

```
10 OPEN 15, 8, 15
20 OPEN 5, 8, 5, "0"
30 FOR L=1 TO 50
40 PRINT# 5, "TEST"
50 NEXT
60 PRINT # 15, "B-W:" 5; 0; 1; 1
70 CLOSE 5: CLOSE 15
```

BLOCK-ALLOCATE

Daha önce belirttiğimiz gibi, Program veya sırasal dosyalar diskete kaydedilirken, DOS kullanılan blokların hesabını tutarak bunun listesini BAM'a (Bant 18, sektör 0) kaydeder. Rastgele erişimle istediğimiz bir bloka kayıt yapınca, bu blokun halen dolu olduğu BAM'da gözükmeyecektir. Dolayısıyla daha sonra normal yöntemle yeni bilgiler SAVE edildiğinde, bunlar BAM'da boş gözüken blokların üstüne kaydedilebilir.

Rastgele erişim yöntemini normal dosyalarla birlikte kullanmak için, programımız serbest blok aramak için BAM'yi kontrol etmeli ve BAM'yi, bu blokları kullandığınızı belirtmek için değiştirmelidir. BAM yenilenince, rastgele dosyalarınız -VALIDATE komutunu kullanmadığınız takdirde (bölüm 3)- emin durumda olacaklardır.

BLOCK-ALLOCATE KOMUTU İÇİN FORMAT:

PRINT # dosya no, "BLOCK-ALLOCATE: "sürücü; bant; blok.

Hangi blokların kullanıma açık olduğunu nasıl anlarsınız? Açık olmayan bir bloğu denersiniz, DOS hata mesajını 65, NO BLOCK'a ayarlayacak ve bant ve blok numaralarını açık olan bir sonraki bant ve blok numaralarına çevirecektir.

Dolayısıyla, diske bir blok yazmaya her kalkıştığınızda, önce o bloğu serbest hale getirmelisiniz. Blok boş değilse, hata kanalından bir sonraki boş bloğu okuyun ve serbest bırakın.

BLOK AYIRMA İŞLEMİ ÖRNEĞİ

```
10 OPEN 15, 8, 15
20 OPEN 5, 8, 5, "#"
30 PRINT # 5, "VERİ"
40 T=1:B=1
50 PRINT # 5, "B-A:" 0;T;B
```

```
60 INPUT # 15, A, B$, C,D
70 IF A=65 THEN T=C:B=D: GOTO 50
80 PRINT # 15, "B-W:" 5; 0; 1; 1;
90 CLOSE 5: CLOSE 15
```

BLOCK-FREE

BLOCK-FREE komutu, BLOCK-ALLOCATE komutunun tam tersidir. Artık kullanmak istemediğiniz blokları, sistem kullanımı için serbest bırakır. Dosyalar için kullanılan SCRATCH komutuna benzer. Ancak diskten veri silmeyip, yalnızca girişi, bu durumda yalnızca BAM'yi serbest bırakır.

BLOCK-FREE KOMUTU İÇİN FORMAT:

PRINT # dosya no, "BLOCK-FREE: "sürücü; bant; blok veya kısaca:
PRINT # dosya no, "B-F:" sürücü; bant; blok

RASTGELE ERİŞİM YÖNTEMİNİN KULLANIMI

Rastgele erişim yöntemi hakkında şimdiye kadar gördüklerimiz içinde tek problem, diskte hangi blokların kullanıldığının hesabını tutamamasıdır. BAM içinde, kullanılan bir blok diğerinden ayırdedilemez. Orada rastgele erişim kaydınızın mı, bir program parçasının mı, hatta sırasal veya relatif dosyaların mı bulunduğunu bilemezsiniz.

Bu hesabı tutmak için, çözüm, rastgele dosya ile birlikte bir sırasal dosya tutmaktır. Bu dosyada yalnızca kayıt, bant ve blok konumları olsun. Buna göre her rastgele dosya için diske açık 3 kanal olacaktır. Bir tane komut kanalı, bir tane rastgele veri için, bir de sırasal veri için. Bu, aynı zamanda doldurduğunuz 2 adet de tampon var demektir.

10 RASTGELE ERİŞİMLİ BLOK YAZAN SIRASAL DOSYALI ÖRNEK PROGRAM:

```
10  OPEN 15, 8, 15
20  OPEN 5, 8, 5, "#"
30  OPEN 4, 8, 4, "@Ø: ANAHTAR, S, W"
40  A$="KAYIT İÇRİĞİ #!"
50  FORR=1 TO10
70  PRINT#! 5, A$","R
90  T=1:B=1
100 PRINT #15, "B-A:" 0; T; B
110 INPUT # 15, A, B$, C, D
130 IF A=65 THEN T=C:B=D: GOTO 110
130 PRINT# 15, "B-W:" 5; 0; T; B
140 PRINT # 4, T","B
150 NEXT TR
160 CLOSE 4: CLOSE 5: CLOSE 15
```

Bir sonraki boş bant ve bloğu bul

10 RASTGELE ERİŞİMLİ BLOĞU SIRASAL DOSYA İLE OKUYAN ÖRNEK PROGRAM:

```
10  OPEN 15, 8, 15
20  OPEN 5, 8, 5, ''#''
30  OPEN 4, 8, 4, ''ANAHTAR, S,R''
40  FOR R=1 TO 10
50  INPUT# 4, T,B
60  PRINT# 15, ''B-R,'' 0;T;B
80  INPUT# 5, A$,X
100 IF A$ < > ''KAYIT İÇERİĞİ #''OR X  < > R THEN STOP
110 PRINT A$; R
120 PRINT# 15, ''B-F:'' 0;T;B
130 NEXT R
140 CLOSE4: CLOSE5
150 PRINT# 15, ''$0:ANAHTAR''
160 CLOSE 15
```

Bant ve bloğu sonradan kullanmak üzere serbest bırak.

BUFFER-POINTER

Tampon göstergesi, son veri yazılan yerin hesabını tutar. Aynı zamanda bir sonraki verinin nereden okunacağını gösterir. Tampon göstergesinin tampon içindeki yerini değiştirerek, bir blok içindeki byte'lara rastgele erişim sağlanabilir. Böylece, her blok alt-kayıtlara bölünebilir.

TAMPON GÖSTERGESİ KOMUT FORMATI:

PRINT# dosya no, ''BUFFER-POINTER: ''Kanal; konum veya kısaca:
PRINT# dosya no, ''B-P:'' kanal; konum

GÖSTERGEYİ TAMPONUN 64'ÜNCÜ KARAKTERİNE GETİRME ÖRNEĞİ:

PRINT # 15, ''B-P:'' 5; 64

Yukarıdaki programların, bloklar içindeki kayıtlara göre çalışan, değişik şekilleri şöyle olabilir:

HER BİRİ 4 KAYIT İÇEREN 10 RASTGELE-ERİŞİMLİ BLOK YAZAN ÖRNEK PROGRAM:

```
10  OPEN 15, 8, 15
20  OPEN 5, 8, 5, ''   ''
30  OPEN 4, 8, 4 ''@0:ANAHTAR,S,W''
40  A$=''KAYIT İÇERİĞİ   ''
50  FOR R=1 TO 10
60  FOR L=1 TO 4
70  PRINT#15, ''B-P:'' CHR$ (5) CHR$((L-1)*64+1)
80  PRINT# 5, AS'',''L
90  NEXT L
100 T=1:B=1
```

konumu 1,65 129, 193'e al

```
110 PRINT # 15, ''B-A:'' 0;T;B
120 INPUT # 15, A, B$, C, D,
130 IF A=65 THEN T=C:B=D: GOTO 110
140 PRINT # 15, ''B-W:'' 5;0;T;B
150 PRINT # 4,T '','' B
160 NEXT
170 CLOSE4: CLOSE5:CLOSE15
```

boş bant ve blok

HER BİRİ 4 KAYIT İÇEREN 10 RASTGELE ERİŞİMLİ BLOĞU OKUYAN ÖRNEK PROGRAM

```
10  OPEN 15, 8, 15
20  OPEN 5, 8, 5, ''   ''
30  OPEN 4, 8, 4, ''ANAHTAR, S,R''
40  FOR R=1 TO 10
50  INPUT#4, T, B
60  PRINT # 15, ''B-R:'' 5; 0; T; B
70  FOR L=1 TO 4
80  PRINT# 15, ''B-P:'' 5; ((L-1)*64+1)
85  INPUT#5, A$,X
90  IF A$ < > ''KAYIT İÇERİĞİ #'' ORX < > L THEN STOP
95  PRINT R; A$; L
100 NEXT L
110 PRINT # 15, ''B-F:'' 0; T; B
120 NEXT R
130 CLOSE 4: CLOSE 5
140 PRINT # 15, ''S0:ANAHTAR''
150 CLOSE 15
```

konumu 1,65, 129, 193'e al

USER1 ve USER2

USER komutları genelde makina dili içindir (daha çok detay için bölüm 8'e bakın). USER 1 ve USER2 komutları BLOCK-READ ve BLOCK-WRITE komutlarının birer versiyonudurlar, şu farkla ki USER1 ve USER2 tampon göstergesi ile çalışırlar.

BLOCK-READ Komutu 256 karaktere kadar okur, ancak tampon işaret edici bloğun bittiğini söyleyince durur. USER1 komutu BLOCK-READ işlemi yapar, ancak diskten bloğun tümünü okumak için göstergeyi önce 255'inci karaktere getirir.

USER1 KOMUT FORMATI

PRINT # dosya no, ''U1:'' kanal; sürücü; bant; blok veya
PRINT # dosya no, ''UA,'' ''kanal; sürücü; bant; blok

Bu komut için U1 ve UA tipleri arasında fark yoktur. BLOCK-WRITE komutu, tamponun içeriğini diskteki bloğa, tampon göstergesinin değeri ile birlikte yazar. USER2 komutu tamponun, disketin o bloğunda kalan mevcut gösterge de-

ğerine dokunmaksızın yazar. Bu, bir blok BLOCK-READ ile okunup, BUFFER-POINTER ve PRINT komutları ile yenilenerek, USER2 ile diskete yeniden yazılacağı zaman, yararlıdır.

USER2 KOMUT FORMATI

PRINT # dosya no, "U2:" kanal; sürücü; bant; blok veya
PRINT # dosya no: "UB:" kanal; sürücü; bant; blok
Daha karmaşık bir örnek program için, ek C'ye bakın.

7. RELATİF DOSYALAR

Relatif dosyalama yöntemi, temelde, bir önceki bölümde anlatılan yöntem gibi çalışır; bir dizi veriyi (örneğin bir müşteri-isim-adres-hesap listesini) disket üzerinde belli bir düzenle kaydederek, istenen noktaya istenildiği gibi ulaşılmasını sağlar. Daha önce anlatılan yöntemin aksine, bu kez hangi kaydın diskette nerede olduğunu hesaplama işini DOS, kendiliğinden yapacaktır.

Bu amaçla, her dosya için DOS bir "yan sektör" (bir çeşit çetele defteri) açar. Yan sektör, dosyadaki her girişin disket üzerindeki yerini DOS'a bildirir. Böylece, örneğin 1000 maddelik bir listeden sadece 473'üncüyü okumak veya değiştirmek mümkün olur. Her yan sektör 120 kayıda kadar işaret edebilir ve bir dosyada 6 yan sektör olabilir. Bir dosyada 720 kayıt ve her kayıtta 254 karakter olabildiği için, bir dosya, tüm bir disketi kaplayabilir.

RELATİF DOSYANIN YARATILMASI

Bir relatif dosya ilk kullanılacağı zaman, OPEN komutu ile yaratılır; sonradan, aynı dosya kullanılır. Değiştirme opsiyonu (@ işareti ile), dosyayı silip yeniden yaratmaz. Dosya defaatle genişletilebilir, okunabilir ve yazıma açıktır.

RELATİF DOSYA YARATACAK OPEN KOMUT FORMATI:

OPEN dosya no, cihaz no, kanal no, "isim, L," + CHR$ ← kayıt uzunluğu

RELATİF DOSYA YARATAN OPEN KOMUT ÖRNEKLERİ:

OPEN 2, 8, 2, "DOSYA, L," + CHR$ (100) ←
OPEN F, 8, F, A$ + "L," + CHR$ (A) kayıt uzunluğu
OPEN A,B,C, "TEST, L," + CHR$ (33)

Dosyadaki her madde (kayıt) standart bir uzunlukta olmak zorundadır. Dosya örneğin bir isim listesi ise, en uzun ismin tahminen 20 karakter olacağı düşünülerek, şöyle dosya açılabilir.

OPEN 2, 8, 2 "LISTE,L," + CHR$ (20)

TABLO 7.1.RELATİF DOSYA FORMATI

VERİ BLOĞU

BYTE	TANIM
0,1	Bir sonraki veri bloğunun bant ve bloğu
2-256	254 veri baytı Dosya açıldığında O'larla doldurulur. Madde sonu 255'le belirtilir.

YAN SEKTÖR BLOĞU

BYTE	TANIM
0,1	Bir sonraki yan sektör bloğunun bant ve bloğu.
2	Yan sektör numarası (0-5)
3	Madde uzunluğu
4,5	İlk yan sektörün bant ve bloğu (0)
6,7	İkinci yan sektörün bant ve bloğu (1)
8,9	Üçüncü yan sektörün bant ve bloğu (2)
10,11	Dördüncü yan sektörün bant ve bloğu (3)
12,13	Beşinci yan sektörün bant ve bloğu (4)
14,15	Altıncı yan sektörün bant ve bloğu (5)
16-256	120 veri bloğu için bant ve blok numaraları.

Çalışmada, DOS önce dosyanın varolup olmadığına bakar. Varsa, birşey olmaz. Eski bir relatif dosyayı silmenin tek yolu SCRATCH (4 bölüm) komutu iledir, değiştirme opsiyonu ile yapılmaz.

RELATİF DOSYALARIN KULLANILMASI

Var olan bir relatif dosyayı açmak için format daha basittir.

MEVCUT BİR RELATİF DOSYAYI AÇMAK İÇİN FORMAT:

OPEN dosya no, cihaz no, kanal no, ''isim''

Bu durumda DOS otomatik olarak bunun bir relatif dosya olduğunu anlar. OPEN komutu, hem dosyaya yazmak hem de dosyadan okumak için geçerlidir. Ancak, okuma ya da yazma için, herhangi bir işlemden önce, dosya pozisyon göstergesini istenen madde numarasına getirmelisiniz.

POSITION KOMUT FORMATI

PRINT # dosya no, "P" CHR$ (kanal no + 96) + CHR$ (madde no alt) + CHR$ (madde no üst) + CHR$ (madde içinde pozisyon)

POSITION KOMUT ÖRNEKLERİ

```
PRINT # 15, "P"CHR$ (2 + 96) CHR$(1) CHR$(0)
PRINT # 15, "P"CHR$ (CH + 96) CHR$(R1) CHR$(R2)
PRINT # 15, "P"CHR$ (4 + 96) CHR$ (R1) CHR$ (R2) CHR$(P)
```

Kayıt no

Kanal no

Kayıt içi konum

Bir byte yalnızca 256 değişik rakam alabildiği ve dosyada 700'den fazla kayıt olabileceği için kayıt numarasını 2 byte'a bölmek gereklidir. Bu: kayıt no = üst kayıt no * 256 + alt kayıt no formülü ile gerçek kayıt no'suna çevrilebilir.

Kayıt numarası bilinmiyorsa, alt ve üst byte'lar şöyle bulunabilir:

Üst kayıt no = INT (kayıt no/256)
Alt kayıt no = kayıt no-üst kayıt no * 256

ÖRNEK:
PRINT #15, "P"CHR$(4 + 96) CHR$(R1) CHR$(R2) CHR$(0)

Eğer kayıt no 540 ise, R2 = INT (540/256)...R2 = 2
RI = 540 - R2 * 256...R1 = 28

Bir posta adresi listeniz ve tabloya göre 8 veri parçası olsun:

Alan İsmi	Uzunluk
İsim	12
Soyadı	15
1. adres	30
2. adres	24
Şehir	12
Telefon No.	7
TOPLAM	100

Kayıt uzunluğu şöyle bulunur: Her alan için fazladan bir karakter uzunluk, ayırma için bırakılmalıdır; yoksa INPUT # komutu, sırasal dosyalardaki gibi, ihtiyaçtan çok daha uzun bir dosya parçasını kaplayacaktır. Böylece, kayıt başına 106 karakterlik bir dosya açarız. İlk kayıda 1 numarasını veririz. Tarif edildiği kadarıyla program şöyledir:

```
5 X = 0
10 OPEN 1, 8, 15
20 OPEN 2, 8, 3, "0:POSTA LİSTESİ, L," + CHR$(106)
30 GOSUB 900
40 PRINT#1, "P"CHR$(3+96) CHR$(1) CHR$(0) CHR$(1)
50 GOSUB 900
60 IF E=50 THEN PRINT#2,1:GOTO 40
70 INPUT#2,X
75 PRINT X
300 STOP:CLOSE1; CLOSE2; END
900 INPUT#1,E,B$,C,D          <-- Hata alt rutini
910 IF (E=50) OR (E<20) THEN RETURN
920 PRINT E;B$;C;D:STOP:RETURN
```

Programda satır 60'da kontrol edilen hata 50 numaralı KAYIT YOK hatasıdır ve kayıt daha yaratılmamış demektir.

Kayıt içine yazmak, problemi çözer. Programlarınızda bu hataya dikkat etmelisiniz.

Şu ana kadar yapılan, dosyanın yaratılması ve ilk kaydının konması idi; ancak daha veriler alınmamıştır. Aşağıda programın, kayıtların rakamlarla gösterildiği bir posta listesine ilişkin, gelişmiş şekli vardır:

```
5 A (1) = 12:(2) = 15:A (3) = 30:A (4) = 24:A (5) = 12:A (6) = 7
10 OPEN 1, 8, 15:OPEN 2, 8, 3, "0:POSTA LİSTESİ, L," +
CHR$ (106): GOSUB 900
20 PRINT #1 "P" CHR$ (3+96) CHR$ (1) CHR$ (0) CHR$ (1): INPUT 2,
X$X=VAL (X$): IFX=0 THEN X=2
30 INPUT "OKU, YAZ VEYA DUR"; J$: IF J$="D" THEN CLOSE 2:
CLOSE:1 END
40 IF J$="Y" THEN 200                          <-- Okuma rutini
 50 PRINT # INPUT "KAYIT #"; R:IF R < 0 OR R>X THEN 50
60 IF R < 2 THEN 30
70 R1=R:R2=0: IF R1 >256 + THEN R2=INT (R1/256): R1=R1—256*R2
80 RESTORE:DATA1,AD,14,SOYAD,30,ADRES1,61,ADRES2
90 DATA86,ŞEHİR,99,TELEFON
100 FOR L=1 TO 8:READ A,A$:PRINT#1,"P"CHR$(3+96) CHR$(R1) CHR$
(R2) CHR$(A) GOSUB 900
110 ON A/50 GOTO 50                            <-- Yazma rutini
115 INPUT# 2,Z$:PRINT A$,Z$:NEXT:GOTO50
200 PRINT:INPUT "KAYIT#";R:IF R<0 OR R> 500 THEN 200
210 IF R< 2 THEN 30
215 IF R>K THEN R=X+1:PRINT:PRINT "KAYIT NO";R
220 R1=R:R2-0:IF R>255 THEN R2=INT (R1/256):R1=R1-256*R2
230 RESTORE: FOR L=1 TO8: READ A, A$:PRINT#1, "P" CHR$ (3+96)
CHR$ (R1) CHR$ (R2) CHR$(A)
232 GOSUB 900
235 PRINT A$:INPUT Z$:IF LEN (Z$)>A(L) THEN Z$=LEFT$ [Z$,A (L)]
240 PRINT#2,Z$:GOSUB 900, NEXT:IF R>X THEN X=R
245 PRINT#1,"P" CHR$(3+96) CHR$(1) CHR$(0) CHR$(1)
```

```
250 PRINT # 2,X:GOSUB 900:GOTO 200
900 INPUT # 1,A,B$,C,D:IF A<20 THEN RETURN
910 IF A< >50 THEN PRINT A,B$,C,D:STOP:RETURN
920 IF J$="0" THEN PRINT B$
930 RETURN
```

215. satır yazılması istenen kayıt numarasını sormaktadır. Verilen no. son yazılan kayıt no'dan büyükse, numara otomatik olarak son yazılan no +1 kabul edilmektedir.

Bundan daha gelişmiş bir program, kayıtları indekslemek üzere, "anahtarlar" kullanarak elemanların hesabını tutmaktadır. Örneğin, bir kaydı isimden aramak, veya etiketleri kod ile yazmak isteyebilirsiniz. Bunun için anahtar ve kayıt numaralarının, muhtemelen sırasal dosyalarda listeleri gereklidir.

Yakında çok büyüyecek olan bir relatif dosyayla çalışırken, dosyanın öngörülen sonuna bir kayıt yazmak iyidir. Örneğin, dosyanın 1000 kayıtlık olacağını öngörüyorsanız, dosyayı yaratır yaratmaz, 1000'inci kayıdı yazın. Bu DOS'un aradaki kayıtları doldurduktan sonra, bunları çok daha hızlı kullanmasını sağlayacaktır.

BÜYÜK DOSYA YARATMA ÖRNEĞİ:

```
OPEN 1,8,15:OPEN 2,8,2 "0,REL,L," +CHR$(60)
PRINT#1, "P" CHR$(2+96) CHR$(0) CHR$(4) CHR$(1)
PRINT#2, "SON"
CLOSE2:CLOSE1
```

PROGRAM VE AÇIKLAMA

```
  1 REM RELATİF DOSYA PROGRAMI
  2 DIM A$(5):DIMC$(5):PRINT "0"
  3 PRINT "RELATİF DOSYA AÇMAK İÇİN F1'E BASIN"
  4 PRINT "RELATİF DOSYA OKUMAK İÇİN F2'YE BASIN"
  5 PRINT "HATA KANALINI OKUMAK İÇİN F5'E BASIN":GOSUB 5000
  6 INPUT "RELATİF DOSYA ADINI GİRİN";ZS
  8 OPEN 2,8,2,Z$+",L," +CHR$(50): REM RELATİF DOSYAYI AÇIN
  9 OPEN 1,8,15
 11 GOSUB 1000
 20 FOR I=1 TO 5
 30 PRINT#1, "P" CHR$(2+96) CHR$(I) CHR$(0) CHR$(1):REM KAYIT
    İŞARET EDİCİYİ KONUMLAYIN
 40 PRINT "BİR İSİM GİRİN"
 50 INPUT A$(I)
 60 PRINT#2 A$(I)
 63 INPUT "EK BİLGİ GİRİŞ";C$(I)
 65 PRINT#1,"P" CHR$(2+96) CHR$(I) CHR$(0) CHR$(25):REM İŞARET
     EDİCİYİ 25'İNCİ KARAKTERE ALIN
 67 PRINT# 2,C$(I)
 70 NEXT I
 75 PRINT "KAYIT DEĞİŞTİRMEK İSTER MİSİNİZ?":INPUT D$
 76 IF D$="N"THEN 80
```

```
  77 GOSUB 8000
  78 GOTO 75
  80 PRINT ''RELATİF DOSYA AÇILDI''
  82 FOR DE=1 TO 2500:NEXT DE:GOSUB 6000
  85 CLOSE 2
  90 END:STOP
100 INPUT ''OKUNACAK DOSYAYI GİRİN'';Z$
105 OPEN 2,8,2,Z$:OPEN 1,8,15
106 PRINT Z$ ''OKUNUYOR''
110 FOR I=1 TO 5
115 REM FOR I=5 TO STEP-1
130 PRINT#1,''P''CHR$(2+96) CHR$(I) CHR$(0) CHR$(1)
160 INPUT#2, A$(I)
170 PRINT ''KAYIT#(''I'')=',A$(I)
175 K=6—1
177 PRINT#1, ''P'' CHR$(2+96) CHR$(I) CHR$(0) CHR$(25)
179 INPUT#2,C$(I):PRINT ''EK BİLGİ:'';C$(I)
180 NEXT I
181 PRINT ''KAYIT DEĞİŞTİRMEK İSTER MİSİNİZ?'':INPUT D$
182 IF D$=''N'' THEN 185
183 GOSUB 8000
184 GOTO 181
185 GOSUB 1000
186 J=I+1
190 PRINT ''OKUMA BİTTİ'':FOR DE=1 TO 1500:NEXT DE: GOSUB
6000:CLOSE 2:CLOSE 1:END
1000 INPUT#1,A,B$,C,D:IF A>20 THEN RETURN
1001 IF A<>50 THEN PRINT A,B$,C,D:STOP:RETURN
1999 END
2000 OPEN 15,8,15
2001 INPUT#15,A,B$,C,D
2002 PRINT A,B$,C,D
2003 CLOSE 15:END
5000 GET S$:IF S$='' '' THEN 5000:REM FONKSIYON TUŞLARININ CHR$
KODLARI İÇİN KLAVYE TARAMASI
5001 IF S$=CHR$(133) THEN 6:REM F1 FONKSİYONU
5002 IF S$=CHR$(134) THEN 100:REM F2 FONKSİYONU
5003 IF S$=CHR$(135) THEN 2000:REM F5 FONKSİYONU
5004 RETURN
6000 PRINT ''KOPYA İÇİN F7'YE, DEVAM İÇİN C'YE BASIN''
6001 GET P$:IF P$<>CHR$(136) AND P$=''C''THEN RETURN
6002 FOR DE=1 TO 500:NEXT DE:IF P$='' '' OR P$<>CHR$(136)
     THEN 6ØØØ
6003 OPEN 4,4:CMD4
6004 PRINT# 4,Z$ ''DOSYANIN İÇERİĞİ''
6005 FOR I=1 TO 5
6010 PRINT#4, ''KAYIT#'';I;'';=A$(I)
6012 PRINT#4, ''EK BİLGİ'';I;''=;C$(I)
6015 NEXT I
```

```
6020 CLOSE 4:RETURN
7000 REM KAYITLARIN TERSTEN OKUNMASI, SATIR 115'TEKİ REM'İ
KALDIRIN VE
7002 REM SATIR 110'DAN ÖNCE REM KOYUN
7005 REM 1'NCİ KAYDIN 1'NCİ ALANI İLE SONUNCUSUNUN 2'NCİ
ALANININ OKUNMASI
7007 REM SATIR 177'DEKİ CHR$(I)'Yİ CHR$(K) İLE DEĞİŞTİRİN
8000 PRINT ''HANGİ KAYIT # DEĞİŞECEK?''; INPUT I
8001 INPUT ''YENİ KAYDI GİRİN'';A$(I)
8002 PRINT#1, ''P'' CHR$ (2+96) CHR$(I) CHR$(0) CHR$(1): REM
DOSYA İŞARET EDİCİYİ KONUMLAYIN
8003 PRINT#2,A$(I)
8004 INPUT ''YENİ KAYDI GİRİN (ALAN 2)'';C$(I)
8005 PRINT#1,''P''CHR$(2+96) CHR$(I) CHR$(0) CHR$(25):REM
DOSYA İŞARET EDİCİYİ KONUMLAYIN
8007 PRINT#2, C$(I)
8009 PRINT ''KAYIT '';I;''DEĞİŞTİRİLDİ''
8010 RETURN
```

RELATİF DOSYA YARAT

Program, 2 alanlı bir relatif dosya yaratmaktadır. İlk alan her kayıdın 1'inci, ikinci alan ise her kayıdın 25'inci pozisyonunda başlamaktadır.

Satır 8: Dosyayı yarat. CHR$(50) ile, 50 karakterlik kayıtlara yer aç. 25i bir dosya olarak kullan, satır 6'da değişkeni gir ve dosya yaratmak için F1'e bas.

Satır 30: Bir kaydın ilk karakter pozisyonuna işaret et. P, disk sürücüye, kayıtta birinci karakter pozisyonuna işaret etmesini söylemektedir.

NOT: CHR$ belirtilmezse, karakter pozisyonu 1 olur.

Satır 65: Bir kaydın 25'inci karakter pozisyonuna işaret et. P, disk sürücüye karşıt gelen pozisyona işaret etmesini söylemektedir.

NOT: POSITION komutu dosyaya hiçbir şey yazmamakta ve ondan hiçbir şey okumamakta, yalnızca kayıtta belirtilen pozisyona işaret etmektedir.

RELATİF DOSYAYA YAZ

Satır 60: Satır 30-CHR$(1) ile belirtildiği gibi, karakter pozisyonu 1'e gelmek için, POSİTİON komutundan sonra PRINT # komutunu kullan.

RELATİF DOSYADAN OKU

Satır 130: POSITION komutu ile, karakter pozisyonunu belirle. Sonra satır 160'taki INPUT # komutu ile devam et.

ÖZET

Bu programda, F1 tuşuna basılınca relatif dosya diskte yaratılmaktadır. RUN yazarak, önceki relatif dosyayı okumak için F3'e basabilirsiniz. Hata durumu olursa (disk sürücüde yanıp sönen kırmızı ışık), RUN/STOP tuşuna basın, RUN yazarak F5'e basabilirsiniz. Bu, sürücüden hata kanalını okuyarak, hata numarasını ve mesajını, ayrıca hatanın bant ve sektörünü yazmaktadır. Sonra sü-

rücüyü kullanım için hazırlar. Daha sonra program, dosyanın yazılıp yazılmayacağını sormaktadır. Dosyayı yazmak için F7'ye, devam etmek için C'ye basın.

UNUTMAYIN:

1) Bir dosyaya yazılmadan veya okunmadan önce bir POSITION komutu verilmelidir.

2) POSITION komutu ile her zaman 15 no.lu kanala açılan dosya no kullanılmalıdır. Bu, verilerinizi komut kanalının kontroluna sokar.

3) POSITION komutundaki ilk CHR$ kodu, veri dosyasını açan OPEN komutundaki üçüncü kod + 96 olmalıdır. (Yukarıdaki programın 8. ve 30. satırlarına bakın.)

4) Kolaylık için, POSITION komutunun ikinci karakter dizisindeki, CHR$ (I), FOR...NEXT döngü değişkenini kullanın. I, kayıt numarasına eşittir (yukarıdaki programın 20. ve 30. satırlarına bakın).

PROGRAMIN DEĞİŞTİRİLMESİ

1) Her kayıda daha fazla alan sokmak için daha çok POSITION komutu ekleyebilirsiniz. Alan eklerken, karakter pozisyon numarasını en son CHR$ olarak eklemeyi unutmayın.

2) Dosyadaki kayıtların uzunluğunu değiştirmek için, satır 8'deki CHR$(50)'nin değerini değiştirin.

3) Dosyadaki kayıt sayısı 30. satırdaki POSITION komutu tarafından tespit edilmektedir. Buradaki örnek 5 kaydı incelemektir; FOR...NEXT döngü değişkeni 1'den 5'e kadardır. Program 720 kayıt inceleyebilir; fakat 512 kayıttan fazlasını incelemek için POSITION komutunun üçüncü CHR$ ( )'i 2 olmalıdır. 257-512 kayıtlar için, bir rakamını kullanın 256 kayıda kadar, FOR...NEXT döngü değişkenini kullanın. ÖRNEK: FOR I = 1TO 256,256 kayıt için yeterlidir.

8. DİSK KONTROL BİRİMİNİN PROGRAMLANMASI

Uzman programcı, disk sürücünün belleğinde duracak ve çalışacak rutinler yazabilir. Disketten okunacak DOS rutinleri de DOS'a eklenebilir. Böylece, disket sürücünün, programçı tarafından tanımlanacak yeni ve ilginç işlemleri yerine getirmesi sağlanabilir.

BLOCK-EXECUTE

Bu komut, makina dilinde bir program içeren bir disketten sürücü belleğine bir blok yükleyerek, tampondaki 0 adresinden itibaren bir RTS komutu gelene kadar, işlenmesini sağlar.

BLOCK-EXECUTE İÇİN FORMAT:

PRINT # dosya no, "B-E:" kanal; sürücü; bant; blok

MEMORY-READ

Disk sürücüde 16K ROM ve 2K RAM vardır. Bunlara veya DOS'un RAM'de oluşturduğu tamponlara, MEMORY komutları ile erişebilirsiniz. MEMORY-READ size; hata kanalından, okunacak byte'ı seçme imkanı verir.

MEMORY-READ İÇİN FORMAT:

PRINT # dosya no, "M-R:"CHR$ (okuma başlangıç adresi alt byte) CHR$ (üst byte) Kanal no 15'den GET # komutu ile okunan bir sonraki byte, disk sürücü belleğindeki o adresten başlayarak takibeden byte'lar sırayla bilgisayara aktaracaktır.

Bu komutu kullanırken hata kanalından alınacak herhangi bir INPUT # tuhaf sonuçlar verecektir. Bu durum diske (bellek komutları haricinde) herhangi bir komut verilerek giderilir.

DİSK KONTROL BELLEĞİNİ OKUYAN PROGRAM:

```
 10 OPEN 15,8,15
 20 INPUT "BELLEK ADRESİ",A
 25 FOR L=1TO 50
 30 A1=INT.(A/256):A2=A-A1*256
 40 PRINT#15, "M-R:" CHR$(A2) CHR$(A1)
 50 GET#15, A$
 60 PRINT ASC ((A$+CHR$(0));
 70 A=A+1
 80 NEXT
 90 INPUT "DEVAM?";A$
100 IF LEFT$(A$,1)="E"THEN 25
110 GOTO 20
```

Hata kanalından karakter al

MEMORY WRITE

MEMORY-WRITE komutu, disk kontrol belleğine bir kerede 34 byte'a kadar yazma imkanı sağlar. MEMORY-EXECUTE ve USER komutları, bu kodun RUN edilmesinde kullanılabilir.

MEMORY-WRITE İÇİN FORMAT:

PRINT # dosya no, "M-W:"CHR$ (yazma başlangıç adresi alt byte) CHR$ (üst byte) CHR$ (veri byte sayısı); CHR$ (veri 1); CHR$ (veri2)...

DİSKE "RTS" YAZMAK İÇİN PROGRAM

```
10 OPEN 8,8,15,"I"
20 PRINT#8, "M-W:" CHR$(0) CHR$(3) CHR$(1) CHR$(96)
30 PRINT#8, "M-E:" CHR$(0) CHR$(3)
40 CLOSE 8
```

Diski başlangıca döndür.

MEMORY-EXECUTE

DOS belleğindeki herhangi bir rutin, RAM veya ROM, MEMORY-EXECUTE komutu ile işletilebilir.

MEMORY-EXECUTE İÇİN FORMAT:

PRINT # dosya no "M-E:" CHR$ (adres alt byte) CHR$ (üst byte) örnek için yukarıdaki satır 30'a bakın.

USER KOMUTLARI

Bölüm 6'da sözedilen USER 1 ve USER 2 komutları ile, bölüm 2'de sözedilen UI+ ve UI- komutlarından başka, USER komutları disk sürücünün RAM belleğindeki bir adres tablosuna Jump komutlarıdır.

USER KOMUT		UFONKSİYON	
U1	veya	UA	Tampon işaret edici değişmeden BLOCK-READ
U2	veya	UB	Tampon işaret edici değişmeden BLOCK-WRITE
U3	veya	UC	$0500'e JMP
U4	veya	UD	$0503'e JMP
U5	veya	UE	$0506'ya JMP
U6	veya	UF	$0509'a JMP
U7	veya	UG	$050C'ye JMP
U8	veya	UH	$050F'e JMP
U9	veya	UI	$FFFA'ya JMP
U;	veya	UJ	
UI+			Commodore 64 hızı
UI-			VIC20 hızı

Bu adreslere, JMP$0520 gibi birer JMP komutu yazarak, diskin belleğinde, kolay kullanımlı bir Jump Table yaratabilirsiniz.

USER KOMUT ÖRNEKLERİ:

```
PRINT#15, "U3"
PRINT#15, "U" + CHR$(50 + 0)
PRINT#15, "UI"
```

9. DİSKET SÜRÜCÜ CİHAZ NUMARASININ DEĞİŞTİRİLMESİ

YAZILIM METODU:

Sürücü cihaz numarasını, donanımdaki bir köprüden alarak RAM içinde saklar. İşlem başlayınca, numaranın yenisiyle değişmesi kolaydır.

CİHAZ NUMARASINI DEĞİŞTİRMEK İÇİN FORMAT:

PRINT # dosya no, "M-W:"CHR$(119) CHR$(0) CHR$(2) CHR$ (adres + 32) CHR$ (adres + 64)

CİHAZ NUMARASI DEĞİŞTİRME ÖRNEĞİ (8'DEN 9'A)

```
10 OPEN 15,8,15
20 PRINT#15, "M-W:" CHR$(119) CHR$(0) CHR$(2) CHR$(9 + 32) CHR$
(9 + 64)
```

Birden fazla sürücünüz varsa, adresi donanımdan değiştirmek (sonraki kısma bakın) daha makuldür. Ancak, zorunluysa, yazılım yönetimi de kolaydır. Sürücülerin her birini ayrı ayrı bağlayıp numaralarını değiştirin. Böylece sürücüler sıkışmaz.

DONANIM METODU

Çoklu sürücü sistemlerinde, sürücünüzün cihaz numarasını kalıcı olarak değiştirmek kolay bir iştir. Gerekli aletler bir tornavida ve bıçaktır.

DONANIM ÜZERİNDE CİHAZ NUMARASI DEĞİŞTİRMEK İÇİN YAPILACAK İŞLER:

1. Cereyan kablosu dahil, sürücüden tüm kabloları sökün.
2. Sürücüyü düz bir yere ters yatırın.
3. Dış kutunun 4 vidasını sökün.
4. Sürücünün sağ tarafını dikkatlice yukarı kaldırıp, üst kasayı çıkarın.
5. Metal kafesin iki yanındaki iki vidayı sökün.
6. Kafesi çıkarın.
7. Cihaz numarası köprülerini bulun. Sürücünüzün yüzü size dönükken, plakanın sol-ortasındadır.
8. 1541 Modeli için 1. ve 2. köprülerden birini veya her ikisini de kesin. 1541 CR Modeli için A ve B köprülerinden birini veya her ikisini kesin. (çizelgeye bakın)
9. Kafesi ve 2 vidası ile, üst kapakla 4 vidasını takın.
10. Kabloları yeniden bağlayın ve cereyanı verin.

CİHAZ NO	KÖPRÜ A/1	KÖPRÜ B/2
8	Kesmeyin	Kesmeyin
9	Kesin	Kesmeyin
10	Kesmeyin	Kesin
11	Kesin	Kesin

10. ASSEMBLY DİLİ VE 1541

1541 disk sürücünüzü, doğrudan assembly diliyle çalıştırmak isterseniz, aşağıdaki bilgileri kullanabilirsiniz.

Aşağıda, her bir kernal rutinin başlangıç bellek konumlarını veren bir dizi alt-rutin vardır. Bu rutinler, bellekteki alt rutin konumuna geçmeyi sağlayan, Assembly komutu JSR ile birlikte kullanılmaktadır.

SUBROUTİNE

SETLFS	=$FFBA	: dosya, cihaz ve kanal no tesbit et
SETNAM	=$FFBD	: dosya adını tesbit et
OPEN	=$FFCO	: dosya aç
CLOSE	=$FFC3	: dosya kapat
CHKIN	=$FFC6	: INPUT kanalı aç
CLRCH	=$FFCC	: tüm kanalları kapat
BASIN	=$FFCF	: dosyadan bir byte al
BSOUT	=$FFD2	: dosyaya bir byte gönder

Her bir rutinin ne yaptığı ve her bir rutine ne gibi parametrelerin girdiği ve çıktığı hakkında daha fazla bilgi için **Commodore 64 Programcının El Kitabı**'-na bakın.

Yukarıdaki alt-rutinlerin bir uygulaması olarak, aşağıdaki program bu rutin-leri, diskten sırasal bir dosya okumada kullanmaktadır. ''TEST'' dosya adının bellekte $C000'da olduğunu farzedin.

```
LDA#$04          : dosya ismi uzunluğu
LDX#$00          : dosya adı, adres alt byte
LDY#$CO          : dosya adı, adres üst byte
JSR SETNAM       : dosya adını tesbit et
LDA#$03          : dosya no
LDX#$08          : cihaz numarası
LDY#$00          : kanal (0=dizi. dosya. oku)
JSR#SETLFS       : dosya, cihaz, kanal no tesbit
JSR OPEN         : dosya aç
LDX#$03          : x-registere dosya no
JSR CHK IN       : INPUT kanalı aç
İŞLEM
JSR BASIN        : bir byte al
BEQ END          : 0 ise dosya sonu veya hata
JSR BSOUT        : karakteri ekrana ver
JMP İŞLEM        : döngü

LDA#$03          : akümülatöre dosya no
JSR CLOSE        : dosyayı kapat
JSR CLRCHN       : kanalları kapat
RTS              : assembly programının sonu
```

EK A: DİSK KOMUTLARININ ÖZETİ:

Genel format: PRINT # dosya #, komut
KOMUT

NEW	''N''
COPY	''C:yeni dosya = eski dosya''
RENAME	''R: yeni ad = eski ad''
SCRATCH	''S:dosya adı''
INTITIALIZE	''I''
VALİDATE	''V''
BLOCK-READ	''B-R:'' kanal; sürücü; bant; blok
BLOCK-WRITE	''B-W:'' kanal; sürücü; bant; blok
BLOCK-ALLOCATE	''B-A:'' sürücü; bant; blok
BLOCK-FREE	''B-F:'' sürücü; bant; blok
BUFFER-POINTER	''B-P:'' kanal; pozisyon
USER1 ve USER2	''Un'':kanal; sürücü; bant; blok
POSITION	''P'' CHR$ (kanal no) CHR$ (kayıt no alt) CHR$ (kayıt no üst) CHR$ (pozisyon)
BLOCK-EXECUTE	''B-E:'' kanal; sürücü; bant; blok
MEMORY-READ	''M-R:'' CHR$ (adres alt) CHR$ (adres üst)
MEMORY-WRITE	''M-W:'' CHR$ (adres alt) CHR$ (adres üst) CHR$ (karakter sayısı) ''veriler''
MEMORY-EXECUTE	''M-E:'' CHR$ (adres alt) CHR$ (adres üst)
USER Komutları	''Un''

EK B: CBM FLOPİ HATA MESAJLARININ ÖZETİ:

0 OK, hata yok.
1 Kaç dosya silindiği. Hata değil.
2-19 Kullanılmayan hata mesajları. Önemsiz.
20 Diskte blok başlığı bulunamadı.
21 Sync. karakteri bulunamadı.
22 Veri bloğu yok.
23 Veri bloğunda toplam hatası.
24 Byte çözümleme hatası.
25 Yazma-doğrulama hatası.
26 Yazma koruma var.
27 Başlıkta toplama hatası.
28 Uzun veri bloğu
29 Disk tanım kodu uyuşmazlığı
30 Genel yazım hatası.
31 Geçersiz komut.
32 Uzun satır.
33 Geçersiz dosya adı.
34 Dosya verilmemiş.
39 Komut dosyası bulunamadı.
50 Kayıt yok.

51 Kayıtta taşma.
52 Dosya çok büyük.
60 Dosya yazıma açık
61 Dosya açık değil.
62 Dosya bulunamadı.
63 Dosya mevcut.
64 Dosya tipi uyuşmazlığı
65 Blok yok.
66 Yanlış bant ve sektör
67 Yanlış sistem bant veya sektörü.
70 Boş kanal yok.
71 Katalog hatası.
72 Disk dolu veya katalog dolu.
73 Yanlış DOS versiyonu.
74 Sürücü hazır değil.

DOS HATA MESAJLARININ AÇIKLAMASI

20: READ ERROR (okuma hatası) (blok başlığı bulunamadı)

Disk kontrol birimi, istenen veri bloğunun başlığını bulamamış demektir. Kuraldışı bir blok numarası vardır veya başlık bozulmuştur.

21: READ ERROR (okuma hatası) (sync. karakteri yok)

Disk kontrol birimi, istenen iz üzerinde bir sync. işareti bulamamaktadır. Disketin mevcut olmaması, okuma/yazma kafasının ayarsız olması, formatlanmamış veya iyi oturmamış disket nedeniyle olabilir. Ayrıca donanımda bir bozukluğa da işaret edebilir.

22: READ ERROR (veri bloğu yok)

Disk kontrol biriminin, iyi yazılmamış bir veri bloğunu okuması veya doğrulaması istenmiştir. Bu hata mesajı BLOK komutlarına göre oluşur ve kural dışı bir iz ve/veya blok isteğine işaret eder.

23: READ ERROR (veri bloğunda toplam hatası)

Bu hata mesajı, veri byte'larında bir veya daha çoğunda bir hata olduğuna işaret eder. Veriler DOS belleğine okunmuş, fakat veri üzerindeki toplam kontrolü hatalıdır. Bu mesaj ayrıca topraklama problemlerine işaret edebilir.

24: READ ERROR (byte çözümleme hatası)

Veriler veya başlık DOS belleğine okunmuş fakat, veri byte'ındaki geçersiz bir bit örneğinden dolayı bir donanım hatası oluşmuştur. Bu mesaj ayrıca topraklama problemlerine işaret edebilir.

25: WRITE ERROR (yazma hatası) (yaz-doğrula hatası)

Bu mesaj, yazılan verilere, DOS belleğindeki veriler arasında bir uyuşmazlık, disk kontrol birimince farkedilirse verilir.

26: WRITE PROTECT ON (Koruma çentiği kapalı)

Kontrol biriminden, yazma koruma çentiği kapalı iken, bir veri bloğu yazması istenirse verilir. Tipik olarak buna neden, çentiğinin üstü kapalı olan bir disket kullanılmasıdır.

27: READ ERROR (başlıkta toplam hatası)

Kontrol birimi, istenen veri bloğunun boşluğunda bir hata bulmuştur. Blok, DOS belleğine okunmamıştır. Bu mesaj ayrıca topraklama problemlerine işaret edebilir.

28: WRITE ERROR (uzun veri bloğu)

Kontrol birimi bir veri bloğunu yazdıktan sonra, bir sonraki başlıkta bir ayraç işareti bulmaya çalışır. Eğer sync. işareti önceden belirli bir süre içinde bulunmazsa, bu mesaj verilir. Hatanın nedeni kötü disket formatı (verimin bir sonraki bloğa taşması) veya bir donanım bozukluğudur.

29: DISK ID MISMATCH (Disk tanım kodu uyuşmazlığı)

Bu mesaj, kontrol biriminden başlangıç değerleri verilmemiş bir disketi okuması istendiğinde verilir. Bu mesaj ayrıca, bir disketin başlığı kötü ise oluşabilir.

30: SYNTAX ERROR (genel yazılım)

DOS, komut kanalına gönderilen komutu anlayamamaktadır. Tipik olarak bunun nedeni, kuraldışı dosya isim sayısıdır. Örneğin bir COPY komutunun solunda iki dosya ismi olmalıdır.

31: SYNTAX ERROR (geçersiz komut)

DOS, komutu tanımamaktadır.

32: SYNTAX ERROR (geçersiz komut)

Komut, 58 karakterden uzundur.

33: SYNTAX ERROR (geçersiz dosya ismi)

Örnek eşleme, OPEN veya SAVE komutunda yanlış kullanılmıştır.

34: SYNTAX ERROR (dosya yok)

Dosya ismi, bir komutta unutulmuş veya DOS bunu tanımamaktadır. Tipik olarak, komutta iki nokta üstüste (:) unutulmuştur.

39: SYNTAX ERROR (geçersiz komut)

Bu hata, komut kanalına (ikincil adres, 15) gönderilen komut, DOS tarafından tanınmayınca oluşabilir.

50: RECORD NOT PRESENT (kayıt yok)

INPUT # ve GET # komutları ile, son kayıdın ötesini okumaya çalışmanın sonucudur. Bu mesaj ayrıca bir relatif dosyada, dosyanın sonundan ötedeki bir kayıda pozisyonlama yapılınca da oluşur. Eğer amaç yeni kayıdın (PrintH komutu ile) eklenmesiyle dosyanın büyütülmesi ile, mesaj gözardı edilir. Bu hata bulunduktan sonra, yeniden pozisyonlama yapılmaksızın INPUT ve GET kullanılmamalıdır.

51: OVER FLOW IN RECORD (kayıtta taşma)

Relatif dosyalarda PRINT # komutu kayıt sınırını aşmaktadır. Bilgi kesilmektedir. Bölme işareti kayıt sonu belirleyici olarak kullanıldığından, kayıt büyüklüğüne dahildir. Bu mesaj kayıttaki toplam karakter sayısı (bölme işareti dahil), belirlenen kayıt uzunluğu aştığında verilir.

52: FILE TOO LARGE (dosya çok büyük)

Bir relatif dosyadaki kayıt pozisyonu, disk taşması olacağını göstermektedir.

60: WRITE FILE OPEN (Yazma dosyası açık)

Bu mesaj, bir yazma dosyası kapanmadan, okuma için açılırsa verilir.

61: FILE NOT OPEN (Dosya açık değil)

Bu mesaj, erişilmek istenen dosya DOS'ta açılmamışsa verilir. Bazen, bu durumda, mesaj verilmez ve istek gözardı edilir.

62: FILE NOT FOUND (dosya bulunamadı)

İstenilen dosya, diskette yoktur.

63: FILE EXISTS (dosya mevcut)

Yaratılmak istenen isimde bir dosya diskette zaten vardır.

64: FILE TYPE MISMATCH (dosya tipi uyuşmazlığı)

Dosya tipi, istenen dosya için, katalog girişindeki dosya tipine uymamaktadır. (Örn: PRG yerine SEQ)

65: NO BLOCK (blok yok)

Bu mesaj, B-A komutu ile birlikte oluşur. Ayrılacak bloğun, daha önce ayrıldığına işaret eder. Parametreler bir sonraki bant ve sektör numarasını verirler. Eğer parametreler sıfır (0) ise, daha yüksek numaralı tüm bloklar kullanıma açık demektir.

66: ILLEGAL TRACK AND SECTOR (yanlış bant ve sektör)

DOS, kullanılan formatta var olmayan bir bant ve sektöre erişmeye çalışmıştır.

67: ILLEGAL SYSTEM T OR S

Bu özel hata mesajı, yanlış bir sistem bant veya sektörüne işaret eder.

70: NO CHANNEL (kanal yok)

İstenen kanal açık değil veya tüm kanallar kullanılmaktadır. DOS üzerinde aynı anda beş sırasal dosya açılabilir. Doğrudan erişim kanalları altı dosya olabilir.

71: DIRECTORY ERROR (katalog hatası)

BAM ile iç sayıcı uyuşmamaktadır. BAM'de ayrılan yerde bir hata vardır; veya BAM, DOS belleğinde fazladan yazılmıştır. Bu problemi düzeltmek için disketi, bellekte BAM'yi yeniden yerleştirmek üzere, INITIALIZE komutunu verin.

Düzeltme nedeniyle bazı aktif dosyalar kapanabilir.

72: DISK FULL (disk dolu)

Disketteki tüm bloklar kullanılmış veya kataloğun giriş limitine gelinmiştir. DISK FULL, 1541 üstünde, açık dosyanın kapatılması için, iki blok varken gönderilir.

73: DOS MISMATCH (73, CMB DOS V2.6 1541)

Bu hata, yanlış bir format ile formatlanmış bir diske yazma istendiğinde verilir. Bu mesaj, sürücü yeni açıldığında da verilebilir.

74: DRIVE NOT READY (sürücü hazır değil)

Sürücüde disket yok veya mandal açık bırakılmış.

NOT: 20'den küçük olan hata mesajları, SCRATCH komutu ile silinen dosya sayısını veren 01 dışında, gözardı edilmelidir.

DİZİ DOSYALAR

1541 El Kitabında Bulunmayan Komutlar:

Commodore 1541 Disk Sürücüsünün İngilizce Kılavuzunda gösterilmeyen, ancak DOS içerisinde mevcut iki komuttan söz etmek istiyoruz.

Birincisi dizi dosyalarla ilgili olan önemli bir komut:

OPEN dosya no, 8, kanal no, ''dizi-dosya adı, S,A''

Dikkat ederseniz, burada dizi dosyaların diğer komutlarından tek fark sonraki A harfidir. Bu komut, daha önce açmış bulunduğunuz ''dizi-dosya adı'' isimli bir dosyaya veri ekleme yapılmasını sağlamaktadır. (A-Add, append). Eğer böyle bir ekleme komutu bulunmasaydı, bir dizi dosyaya ekleme yapabilmek için o dosyayı tümüyle hafızaya almak, veriyi eklemek ve yeni bir dosya halinde diskete yazmak gerekecekti. Ancak yukarıdaki ekleme komutuyla bu işlem tek hamlede yapılabilmektedir.

Bir küçük örnek verelim:

```
10 OPEN 2,8,2,''DENEME,S,W''
20 PRINT#2,''1.KAYIT''
30 CLOSE 2
40 OPEN 2,8,2,''DENEME,S,A''
```

```
50 PRINT#2,''2.KAYIT''
60 PRINT#2,''3.KAYIT''
70 CLOSE 2
```

10 ile 30'uncu satırlar arasında, önce DENEME adındaki dizi dosya açılıyor ve ''1.KAYIT bu dosyaya yazılıp dosya kapatılıyor. 40. satırdan başlayarak yine DENEME dosyasına 2. KAYIT ve 3. KAYIT ekleniyor. Dikkat ediniz, bir dosyaya A komutuyla ekleme yapabilmek için daha önceden bu dosyayı W komutuyla açmamız gerekmektedir.

Yukarıdaki satırları girdikten sonra, bu dosyaya yazdıklarınızı, aşağıdaki satırlarla okuyabilirsiniz.

```
100 OPEN 2,8,2,''DENEME,S,R''
110 FOR I=1 TO 3
120 İNPUT#2,A$
130 PRINT A$
140 NEXT
150 CLOSE 2
```

Kılavuzda bulunmayan ancak gerçekte var olan ikinci komut ise, diskette herhangi bir nedenle CLOSE komutuyla kapatılmamış dizi-dosyaların okunabilmesini sağlamaktadır. Örneğin, diskte açılmış bir dosya ile çalışırken, bir elektrik kesilmesi olduğunda, bu komutu kullanarak açık kalmış dosyayı tekrar okuyabilirsiniz. Bu komutun formatı şöyledir:

OPEN dosya no, 8, kanal no, ''dizi-dosya adı, S,M''

Dizi dosyaları normal olarak okurken kullanacağınız R komutu yerine burada M yer almaktadır. Açık kalmış bir dizi-dosyayı R komutu ile okumaya çalışırsanız;

WRITE FILE OPEN

mesajını alırsınız.

Aşağıdaki program açık kalmış olan HATALI DOSYA'yı okuyup ''YENİ DOSYA'' adıyla diskete yazmakta ve hatalı dosyayı disketten silmektedir.

```
10 OPEN 2,8,3,''HATALI DOSYA,S,M''
20 OPEN 3,8,4,''YENİ DOSYA,S,W''
30 INPUT#2,A$
40 PRINT#3,A$
50 IF ST<>64 THEN 30
60 CLOSE 2: CLOSE 3
70 OPEN 5,8,15,''S:HATALI DOSYA''
80 CLOSE 5
```

NOT: Bu bölüm, DERYA KÖROĞLU tarafından hazırlanmış olup, COMMODORE 1541 Disk Sürücüsü orijinalinde bulunmamaktadır.